

# MPLS-based Recovery

Karen E. Müller



THESIS PRESENTED IN PARTIAL FULFILMENT  
OF THE REQUIREMENTS FOR THE DEGREE OF  
MASTER OF SCIENCE  
AT THE UNIVERSITY OF STELLENBOSCH

Prof. A. E. Krzesinski  
December 2003

# Declaration

I, the undersigned, hereby declare that the work contained in this thesis is my own original work and that I have not previously in its entirety or in part submitted it at any university for a degree.

Signature:

Date:

# Abstract

MPLS-based recovery is intended to effect rapid and complete restoration of traffic affected by a fault in a Multiprotocol Label Switching (MPLS) network. Two MPLS-based recovery models have been proposed: *IP re-routing* which establishes recovery paths on demand, and *protection switching* which works with pre-established recovery paths. IP re-routing is robust and frugal since no resources are pre-committed but it is inherently slower than protection switching which is intended to offer high reliability to premium services where fault recovery takes place at the 100 ms time scale.

This thesis presents an overview of various recovery techniques and addresses the problem of how to find an in some sense optimal set of pre-established traffic engineered recovery paths, given a network with link capacities and traffic demands.

We present and motivate our choice of a nonlinear objective function and optimization method for finding traffic engineered working and recovery paths. A variant of the flow deviation method is used to find and capacitate a set of optimal label switched paths. We present and evaluate two simple methods for computing a set of pre-established traffic engineered recovery paths by using the flow deviation method.

# Opsomming

MPLS-gebaseerde herstel is daarop gemik om verkeer wat deur 'n fout in 'n Multiprotokol Etiketwisseling (*Multiprotocol Label Switching*) (MPLS) netwerk geaffekteer is, vinnig en volledig te herstel. Twee MPLS-gebaseerde herstelmodelle is voorgestel: Internetprotokol-herroetering (*IP re-routing*) wat herstelpaaie op aanvraag tot stand bring, en beskermingssoorskakeling (*protection switching*) wat met voorafbeplande herstelpaaie werk. IP-herroetering is robuust en voordelig aangesien geen netwerkbronne vooraf gereserveer word nie, maar dit is inherent stadiger as beskermingssoorskakeling wat veronderstel is om 'n hoë graad van betroubaarheid aan belangrike dienste te bied waar die herstel van foute in die 100 ms tydskaal plaasvind.

Hierdie tesis verskaf 'n oorsig oor verskeie hersteltegnieke en ondersoek die probleem hoe om 'n optimale versameling van voorafbeplande herstelpaaie te vind, gegee 'n netwerk met skakelkapasiteite (*link capacities*) en verwagte netwerkverkeer.

Ons stel voor en motiveer ons keuse van 'n nie-lineêre objekfunksie en optimeringsmetode om verkeersontwerpde (*traffic engineered*) aktiewe en herstelpaaie te vind. 'n Variant van die vloeideviasie (*flow deviation*)-metode word gebruik om 'n optimale versameling van etiketwisseling (*label switched*) paaie te vind en om 'n optimale hoeveelheid kapasiteit aan die paaie toe te ken. Ons stel voor en evalueer twee eenvoudige metodes om 'n versameling van optimale voorafbeplande herstelpaaie te bereken deur die vloeideviasie-metode toe te pas.



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Multiprotocol Label Switching . . . . .	1
1.2	MPLS-based Recovery . . . . .	2
1.3	The Problem Addressed in this Thesis . . . . .	3
1.4	Structure of the Thesis . . . . .	3
<b>2</b>	<b>A Survey of Recovery Techniques</b>	<b>5</b>
2.1	Framework for MPLS-based Recovery . . . . .	5
2.1.1	Motivation for MPLS-based Recovery . . . . .	5
2.1.2	The Objectives of MPLS-based Recovery . . . . .	6
2.1.3	Recovery Models for MPLS-based Recovery . . . . .	6
2.1.4	Comparison Criteria . . . . .	7
2.2	Network Survivability Layer Considerations . . . . .	8
2.3	Restoration Network Design . . . . .	10
2.4	Traffic Engineering Recovery approaches . . . . .	11
2.4.1	Virtual Path Routing for Survivable ATM Networks . . . . .	11
2.4.2	Enhancing the self-healing capability of statically protected ATM networks . . . . .	12
2.4.3	Enhancements to Traffic Engineering for MPLS . . . . .	12

<i>Contents</i>	vi
2.4.4 Protection through Thrifty Configuration . . . . .	13
2.4.5 Optical Network Restoration . . . . .	14
2.4.6 Shared Backup LSP Restoration . . . . .	14
2.4.7 Fast Reroute Restoration . . . . .	15
<b>3 Linear and Nonlinear Programming Solutions for MPLS Routing</b>	<b>17</b>
3.1 A Linear Optimization Method . . . . .	17
3.1.1 Optimization Criteria . . . . .	18
3.1.2 Linear Program (LP) Solutions . . . . .	20
3.2 A Nonlinear Optimization Method . . . . .	22
3.2.1 Feasibility and Optimality . . . . .	23
3.2.2 The Penalty Function . . . . .	23
3.2.3 Behavior under Light and Heavy Load . . . . .	25
3.2.4 The Flow Deviation Algorithm . . . . .	26
3.3 Experimental results . . . . .	30
3.3.1 Link utilization results from the LP solution and the Flow Deviation program	30
3.3.2 Paths derived from the LP solution compared to paths found with the Flow Deviation program . . . . .	31
3.4 Conclusion . . . . .	36
<b>4 Computing a set of recovery paths: the Healing method</b>	<b>37</b>
4.1 Flow Deviation Algorithms . . . . .	37
4.2 Computing recovery pathsets . . . . .	38
4.3 Explanation of the different experiments . . . . .	39
4.4 Experimental results . . . . .	41

<i>Contents</i>	vii
4.4.1 Comparison of the active paths before and after a link failure . . . . .	41
4.4.2 Performance of the healed recovery paths . . . . .	42
4.5 Conclusion . . . . .	43
<b>5 Computing a set of recovery paths: the <math>\mathcal{B}^{\mathcal{F}}</math> method</b>	<b>50</b>
5.1 Computing the set of $\mathcal{B}^{\mathcal{F}}$ recovery paths . . . . .	50
5.2 Characteristics of our protection model . . . . .	51
5.3 Experimental results . . . . .	51
5.4 Conclusion . . . . .	62
<b>6 Conclusion</b>	<b>63</b>
<b>A Network models</b>	<b>64</b>
<b>B Route degeneracy</b>	<b>68</b>
B.1 Path sets and route degeneracy . . . . .	68
<b>C List of Acronyms</b>	<b>70</b>
<b>Bibliography</b>	<b>72</b>

# List of Tables

4.1	Results of experiment 1 for the 50-node network . . . . .	46
4.2	Results of experiment 2 for the 50-node network . . . . .	46
4.3	Results of experiment 3 for the 50-node network . . . . .	47
4.4	Results of experiment 4 for the 50-node network . . . . .	47
4.5	Results of experiment 1 for the 100-node network . . . . .	48
4.6	Results of experiment 2 for the 100-node network . . . . .	48
4.7	Results of experiment 3 for the 100-node network . . . . .	49
5.1	Number of paths in $\mathcal{A}$ , $\mathcal{A}_i$ , $\mathcal{B}_i$ and $\mathcal{B}^{\mathcal{F}}$ for single link failure scenarios (uni- and bi-directional). . . . .	52
5.2	Percentage of active paths broken and percentage of total offered traffic affected by a single link failure scenario (uni- and bi-directional). . . . .	53
5.3	Commonality among the paths and path bandwidths used before and after uni-directional link failures in the 20-node network . . . . .	54
5.4	Commonality among the paths and path bandwidths used before and after bi-directional link failures in the 20-node network . . . . .	55
5.5	Commonality among the paths and path bandwidths used before and after uni-directional link failures in the 50-node network . . . . .	56
5.6	Commonality among the paths and path bandwidths used before and after bi-directional link failures in the 50-node network . . . . .	57



<i>Contents</i>	ix
5.7 Commonality among the paths and path bandwidths used before and after uni-directional link failures in the 100-node network . . . . .	58
5.8 Commonality among the paths and path bandwidths used before and after bi-directional link failures in the 100-node network . . . . .	59
5.9 Commonality among the paths and path bandwidths used before and after uni-directional link failures in the planar 100-node network . . . . .	60
5.10 Commonality among the paths and path bandwidths used before and after bi-directional link failures in the planar 100-node network . . . . .	61
5.11 The value of the objective function before and after the worst case single link failure scenarios (uni- and bi-directional). . . . .	62



# List of Figures

3.1	Examples of the penalty function . . . . .	25
3.2	10-node utilization bar charts . . . . .	32
3.3	20-node utilization bar charts . . . . .	33
3.4	100-node utilization bar charts . . . . .	34
3.5	10-node: (a) paths and (b) path flows . . . . .	35
3.6	20-node: (a) paths and (b) path flows . . . . .	35
3.7	100-node: (a) paths and (b) path flows . . . . .	35
4.1	KSP path discrimination . . . . .	39
4.2	Experiment 1: (a) paths and (b) path bandwidths for the 50-node network . . . . .	44
4.3	Experiment 2: (a) paths and (b) path bandwidths for the 50-node network . . . . .	44
4.4	Experiment 3: (a) paths and (b) path bandwidths for the 50-node network . . . . .	44
4.5	Experiment 4: (a) paths and (b) path bandwidths for the 50-node network . . . . .	44
4.6	Experiment 1: (a) paths and (b) path bandwidths for the 100-node network . . . . .	45
4.7	Experiment 2: (a) paths and (b) path bandwidths for the 100-node network . . . . .	45
4.8	Experiment 3: (a) paths and (b) path bandwidths for the 100-node network . . . . .	45
5.1	Commonality among the (a) paths and (b) path bandwidths used before and after a worst case uni-directional link failure in the 20-node network . . . . .	54

5.2	Average and standard deviation of the commonality among the (a) paths and (b) path bandwidths used before and after 102 uni-directional link failure scenarios in the 20-node network . . . . .	54
5.3	Commonality among the (a) paths and (b) path bandwidths used before and after a worst case bi-directional link failure in the 20-node network . . . . .	55
5.4	Average and standard deviation of the commonality among the (a) paths and (b) path bandwidths used before and after 51 bi-directional link failure scenarios in the 20-node network . . . . .	55
5.5	Commonality among the (a) paths and (b) path bandwidths used before and after a worst case uni-directional link failure in the 50-node network . . . . .	56
5.6	Average and standard deviation of the commonality among the (a) paths and (b) path bandwidths used before and after 202 uni-directional link failure scenarios in the 50-node network . . . . .	56
5.7	Commonality among the (a) paths and (b) path bandwidths used before and after a worst case bi-directional link failure in the 50-node network . . . . .	57
5.8	Average and standard deviation of the commonality among the (a) paths and (b) path bandwidths used before and after 101 bi-directional link failure scenarios in the 50-node network . . . . .	57
5.9	Commonality among the (a) paths and (b) path bandwidths used before and after a worst case uni-directional link failure in the 100-node network . . . . .	58
5.10	Average and standard deviation of the commonality among the (a) paths and (b) path bandwidths used before and after 236 uni-directional link failure scenarios in the 100-node network . . . . .	58
5.11	Commonality among the (a) paths and (b) path bandwidths used before and after a worst case bi-directional link failure in the 100-node network . . . . .	59
5.12	Average and standard deviation of the commonality among the (a) paths and (b) path bandwidths used before and after 116 bi-directional link failure scenarios in the 100-node network . . . . .	59
5.13	Commonality among the (a) paths and (b) path bandwidths used before and after a worst case uni-directional link failure in the planar 100-node network . . . . .	60

5.14	Average and standard deviation of the commonality among the (a) paths and (b) path bandwidths used before and after 470 uni-directional link failure scenarios in the planar 100-node network . . . . .	60
5.15	Commonality among the (a) paths and (b) path bandwidths used before and after a worst case bi-directional link failure in the planar 100-node network . . . . .	61
5.16	Average and standard deviation of the commonality among the (a) paths and (b) path bandwidths used before and after 235 bi-directional link failure scenarios in the planar 100-node network . . . . .	61
A.1	The 10-node network . . . . .	65
A.2	The 20-node network . . . . .	66
A.3	The 50-node network . . . . .	66
A.4	The 100-node network . . . . .	67
A.5	The planar 100-node network . . . . .	67
B.1	The “fish” network . . . . .	68

# Acknowledgements

This work was performed within the *Siemens-Telkom Centre of Excellence for ATM & Broadband Networks and their Applications* and was supported by grants from the South African National Research Foundation, Telkom SA Limited and Siemens Telecommunications.



# Chapter 1

## Introduction

This chapter explains what *Multiprotocol Label Switching (MPLS)* is and describes the context in which this thesis will discuss MPLS-based recovery in IP networks, followed by a brief outline of the chapters to follow.

### 1.1 Multiprotocol Label Switching

*Multiprotocol Label Switching (MPLS)* is a label switching protocol introduced by the Internet Engineering Task Force (IETF) and the architecture is specified in [17, 48]. MPLS was primarily developed for Internet Protocol (IP) networks, but as the name indicates, its techniques are applicable to any network layer protocol. In this thesis, we focus on the use of IP as the network layer protocol.

In conventional IP routing, the IP header analysis is performed at each hop of the packet's path in the network. In the MPLS forwarding paradigm, the IP header analysis is performed once at the ingress (or source) of the *Label Switched Path (LSP)*. The packets that are forwarded via the same next hop are grouped into a *Forwarding Equivalence Class (FEC)* based on one or more parameters such as the address prefix, the host address, and the Quality of Service (QoS).

The FEC to which the packet belongs is encoded as a short fixed length value known as a *label*. When the packet is forwarded to its next hop, the label is sent along with the packet. During subsequent hops, there is no further analysis of the packet's network layer header. Rather, the label is used as an index into a table, which specifies the next hop and label. The old label is replaced with this new label, and the packet is forwarded to its next hop. This forwarding paradigm is referred to as *label swapping*. A router which supports MPLS is known as a *Label Switching Router (LSR)*.

Labels usually have a local significance and are used to identify FECs based on the type of the un-



derlying network. For instance in Asynchronous Transfer Mode (ATM) networks, the Virtual Path Identifier (VPI) and Virtual Channel Identifier (VCI) are used in deriving the label. Similarly, in Frame Relay networks, the Data Link Control Identifier (DLCI) is used to derive the label.

*Signalling protocols* are used to distribute label information to establish an LSP. The MPLS architecture does not assume that there is a single label distribution protocol. A number of different label distribution protocols are being standardized. Existing protocols have been extended so that label distribution can be piggybacked on them (see e.g. Border Gateway Protocol (BGP), ReSerVation Protocol (RSVP) [10], with its traffic engineering extension, RSVP-TE [5, 20]). New protocols have also been defined for the explicit purpose of distributing labels (see e.g. the Label Distribution Protocol (LDP), together with its extension, Constraint-based Routing LDP (CR-LDP) [23, 27]). These protocols establish LSPs either by calculating the path at the source node and explicitly routing the setup packets, or by doing routing on a per-hop basis, wherein each router determines the next route along the paths. RSVP-TE and CR-LDP also contain fields describing LSP bandwidth requirements which are used to ensure that sufficient bandwidth is available along the path. RSVP uses its own application-level protocol (over IP) for transportation of its messages, while CR-LDP employs TCP.

One of the most important features of MPLS is its support for explicit routes which allows routes to be based on administrative policies, and allows the routes that LSPs take to be carefully designed to allow traffic engineering [4, 6, 36]. Other important benefits of MPLS include multiprotocol support, link layer independence, improved performance due to simplified packet forwarding, aggregation of multiple streams within Layer 2, scalability of network layer routing, support for multiple types of traffic, and enabling different applications with different quality of service [1].

MPLS is likely to be the technology of choice in the future IP-based transport network and therefore MPLS recovery mechanisms for sustaining traffic flows in the event of network equipment failure need to be investigated.

## 1.2 MPLS-based Recovery

MPLS-based recovery is intended to effect rapid and complete restoration of traffic affected by a fault in an MPLS network.

An IETF framework for MPLS-based recovery is described in [25] and discussed in the following chapter. The framework provides a detailed taxonomy of recovery terminology, and it discusses the motivation for, the objectives of, and the requirements for MPLS-based recovery. The principles of MPLS-based recovery are outlined, and comparison criteria that may serve as a basis for comparing



and evaluating different recovery schemes are provided.

Two MPLS-based recovery models have been proposed: *IP re-routing* which establishes recovery paths on demand, and *protection switching* which works with pre-established recovery paths. IP re-routing is robust and frugal since no resources are pre-committed but is inherently slower than protection switching which is intended to offer high reliability to premium services where fault recovery takes place at the 100 ms time scale.

### 1.3 The Problem Addressed in this Thesis

This thesis presents a model of protection switching in MPLS networks. The problem addressed is how to find an in some sense optimal set of pre-established traffic engineered recovery paths, given a network with link capacities and traffic demands.

Another important problem is given a set of working and recovery paths, how to find a set of working paths which will carry the given traffic demands in such a manner, that should a failure occur, recovery mechanisms can quickly and effectively restore traffic flows.

The path discovery procedure is formulated as a constrained, nonlinear optimization problem. An appropriate objective function and optimization method have to be formulated and used to find traffic engineered LSPs.

We focus on recovery after single link failures (uni- and bi-directional) and assume that the networks we work with are sufficiently dimensioned (capacitated) that recovery is possible.

### 1.4 Structure of the Thesis

Chapter 2 describes the framework for MPLS-based recovery and provides an overview of different recovery techniques.

Chapter 3 presents and motivates our choice of a nonlinear objective function and optimization method for finding traffic engineered working and recovery paths. We also investigate a linear objective function which can be used to find traffic engineered working paths such that the spare capacity on each link is maximized. We compare the distribution of the link utilizations of the linear and nonlinear methods and we see that the nonlinear objective function can be parameterized so that it maximizes the spare capacity on each link. The more spare capacity available on links, the easier the recovery mechanisms can restore the network throughput after a failure.

Chapter 4 examines the effectiveness of using healed working paths as recovery paths for a single uni-directional link failure. A healed path deploys a short detour around the failed link without causing any loops in the path.

Finally, chapter 5 presents and evaluates a method for finding traffic engineered working and recovery paths for a given set of possible failure scenarios.

## Chapter 2

# A Survey of Recovery Techniques

This chapter describes the framework for MPLS-based recovery and provides an overview of different recovery techniques.

### 2.1 Framework for MPLS-based Recovery

This section lists the important features of MPLS-based recovery [25] and the criteria which are used to evaluate various MPLS recovery schemes.

#### 2.1.1 Motivation for MPLS-based Recovery

MPLS-based protection of traffic (called MPLS-based Recovery) is useful because it increases network reliability by enabling a faster response to faults than is possible with traditional Layer 3 (IP layer) approaches alone.

The need for MPLS-based recovery arises because of the following: (1) Layer 3 or IP rerouting may be too slow for a core MPLS network that needs to support high reliability and availability, (2) Layer 0 and Layer 1 mechanisms may not be deployed in topologies that meet the carriers' protection goals, (3) the granularity at which the lower layers may be able to protect traffic may be too coarse for traffic that is switched using MPLS-based mechanisms, (4) Layer 0 and Layer 1 may have no visibility into higher layer operations which will prevent the fast restoration of traffic transported at Layer 3, (5) failure scenarios cause transient instability in SPF (*shortest path first*) routing, but an LSP is not affected by the transient instability because source routing is used, and (6) establishing interoperability of protection mechanisms between routers from different vendors in IP or MPLS networks is desired to enable recovery mechanisms to work in a multivendor environment and to enable the transition of



certain protected services to an MPLS core.

### 2.1.2 The Objectives of MPLS-based Recovery

MPLS-based recovery mechanisms and techniques should: (1) be subject to the traffic engineering (TE) goal of optimal use of resources, (2) facilitate restoration times that are sufficiently fast for the end-user applications, (3) maximize network reliability and availability and minimize the number of single points of failure in the MPLS protected domain, (4) enhance the reliability of the protected traffic while minimally or predictably degrading the traffic carried by the diverted resources, (5) protect the traffic at various granularities<sup>1</sup>, (6) be applicable to an entire end-to-end path or to segments of an end-to-end path, (7) take into consideration the recovery actions of the lower layers and should not trigger lower layer protection switching, (8) minimize the loss of data and packet re-ordering during recovery operations, (9) minimize the state overhead incurred for each recovery path maintained and, (10) preserve the constraints on traffic after switchover, if desired, so that the recovery path meets the resource requirements of the working path and achieves the same performance characteristics as the working path.

Some of the above goals are in conflict with each other and the deployment of MPLS-based recovery will involve compromises based on a variety of factors such as cost, end-user application requirements, network efficiency, and revenue considerations.

### 2.1.3 Recovery Models for MPLS-based Recovery

Two recovery models have been proposed for MPLS networks: *IP re-routing* which establishes recovery paths on demand, and *protection switching* which works with pre-established recovery paths.

IP re-routing is robust and frugal since no resources are pre-committed but is inherently slower than protection switching which is intended to offer high reliability to premium services where fault recovery takes place at the 100 ms time scale.

A recovery path may support the same traffic contract as the working path, or it may not. An *equivalent recovery path* can replace a working path without degrading service. A *limited recovery path* lacks the resources (or the resource reservations) to replace the working path without degrading service.

There are two options for the initiation of resource allocation: *pre-reserved* allocation which only applies to protection switching and *reserved-on-demand* allocation which may apply either to IP re-

<sup>1</sup>Three levels of traffic granularity are proposed: part of the recovery traffic can be allocated to an individual path, all of the recovery traffic can be allocated to an individual path, or all of the recovery traffic can be allocated to a group of paths.



routing or to protection switching. A pre-reserved recovery path reserves required resources on all the hops along its route during its establishment before any failure has occurred. A reserved-on-demand recovery path reserves required resources after a failure on the working path has been detected and before the traffic on the working path is switched over to the recovery path(s).

In *dedicated protection*, each working path has one pre-reserved recovery path. The advantage of dedicated protection is its simplicity and fast operation. Its drawback is its high resource usage, especially when the network has to be able to survive multiple ( $n$ ) failures. In this case up to  $n + 1$  independent paths with allocated capacities should be dedicated to each node-pair.

In *shared protection*, working paths that are not expected to fail simultaneously share resources allocated for protection. The advantage of shared protection is the ability to survive multiple failures while the resource usage is moderate. Determining what resources can be shared can be accomplished by offline analysis or by the techniques described in [32] (see also section 2.4.6).

The following terminology is also used to describe dedicated and shared protection:  $1 + 1$  *protection* implies that all data are sent on two paths simultaneously,  $1 : 1$  *protection* implies the dedicated protection technique while  $1 : n$  and  $m : n$  *protection* implies shared protection techniques where 1 and  $m$  protection facilities are available for  $n$  working facilities respectively.

A *static* recovery model assigns a fixed working path and one or more protection paths to each node pair. A *dynamic* recovery model reconfigures the active and recovery paths from time to time. An *adaptive* recovery model determines the protection paths on-the-fly. Adaptive protection is the slowest and needs the most processing, but it does not allocate resources in advance.

*Diverse protection* implies that the working and protection (recovery) paths use diverse, independent paths. This ensures that no single failure can affect both the working and the protection paths. For example, routes can be chosen to be link-disjoint or both link- and node-disjoint.

#### 2.1.4 Comparison Criteria

Several criteria have been suggested for comparing various MPLS-based recovery schemes. The *recovery time* is the time between a failure of a node or link and the time before a recovery path is installed and the traffic starts flowing on it. The *full restoration time* is the time required for traffic to be routed onto links which are capable of (or have been engineered to) handle traffic in recovery scenarios. The full restoration time may differ from the recovery time depending on whether equivalent of limited recovery paths are used. The *setup vulnerability time* is the time that a working path or a set of working paths is left unprotected during such tasks as recovery path computation.



Recovery schemes may require differing amounts of *back-up capacity* in the event of a fault. This capacity will depend on the traffic characteristics of the network. However, it may also depend on the protection plan selection algorithms as well as the signalling and re-routing methods. Recovery schemes may introduce *additive latency* to traffic. For example, a recovery path may take more hops than the working path. This may be dependent on the recovery path selection algorithms. The *quality of protection*: recovery schemes can offer a spectrum of packet survivability options which may range from relative to absolute. Relative survivability may mean that the protected traffic is on an equal footing with other traffic for the surviving network resources. Absolute survivability may mean that the survivability of the protected traffic has explicit guarantees. Recovery schemes may introduce *re-ordering* of packets since the action of putting traffic back on preferred paths might cause packet re-ordering. As the number of recovery paths in a protection plan grows, the *state overhead* required to maintain them also grows. Recovery schemes may require differing numbers of paths to maintain certain levels of coverage. The state overhead may depend on the recovery scheme. In many cases the state overhead will be in proportion to the number of recovery paths. Recovery schemes may introduce a certain amount of *packet loss* during switchover to a recovery path. In the case of link or node failure a certain packet loss is inevitable.

Recovery schemes may offer various types of *failure coverage*. A recovery scheme may account for only certain *types of faults* such as link faults or both node and link faults. The recovery scheme may also respond to service degradation. A recovery scheme may be able to handle *concurrent faults*: depending on the layout of the recovery paths in the protection plan, multiple-fault scenarios may be able to be restored. A recovery scheme can offer *multiple recovery paths*: for a given fault, there may be one or more recovery paths. A recovery scheme may offer a varying degree of *coverage*: depending on the recovery scheme and its implementation, a certain percentage of link and node faults may be covered. Finally, a recovery scheme has a *reaction time*: the number of protected paths may affect how fast the total set of paths affected by a fault can be recovered.

## 2.2 Network Survivability Layer Considerations

Network survivability refers to the capability of the network to maintain service continuity in the presence of faults within the network [4]. Survivability capabilities are available at multiple layers, allowing for protection and restoration to occur at any layer of the network. The advantages and current limitations of network survivability at different layers of the network are examined in [44] and discussed in this section.

The advantages of network restoration at the *optical layer* are fast failure detection and the ability



to restore large numbers of higher layer flows without the need to invoke higher layer signalling. Some current limitations are a limited range of granularity (traffic is restored at lightpath granularity but *individual* circuits or paths are not restored), discrimination between different traffic types is not possible, and the speed of detection is dependent on the locality of the switching action.

The advantages of restoration at the *SONET/SDH layer* are that SONET protection is standardized and can operate across domains, it provides both detection and automatic protection switching, and it provides greater control over the granularity of the channels that can be protection switched. SONET protection is largely limited to ring topologies. Ring topologies hinder the deployment of potentially more efficient, mesh-based restoration schemes, and make inefficient use of spare capacity. The SONET layer cannot distinguish between different priorities of traffic and it is oblivious to higher layer failures.

The advantages of restoration at the *ATM layer* and *MPLS*, are that faults in a router or switch, which are invisible to lower layers, as well as node or software misconfigurations, can be detected. The ATM layer has functionality that can help to detect path errors along a virtual circuit or virtual path, and also provides faster detection and restoration than is possible by relying on routing protocols alone.

The *IP layer* and *Transport layers* are central to the IP network infrastructure. Some IP layer advantages for survivability include the ability to find optimal routes, the ability to provide a fine level of protection granularity, and the ability to perform load sharing by distributing traffic across different paths. Advantages of the Transport layers for survivability include the ability to provide positive acknowledgements with retransmission (ACK) and the ability to provide the finest protection granularity. The drawbacks of both the IP layer and Transport layers in terms of survivability are that connectionless recovery is quite slow relative to the lower layers, and physical layer faults cannot be detected. One of the major considerations for the IP and Transport layer is the time required to detect faults. In order for these layers to provide reliable operation and fast recovery they have to work in conjunction with a path pinning mechanism such as MPLS.

A protection mechanism at different layers (for example, the optical layer and MPLS) could enable IP traffic to be put directly over Wavelength Division Multiplexing (WDM) optical channels, without an intervening SONET layer, thereby facilitating the construction of IP-over-WDM networks.

An important aspect of multi-layer survivability is that the various technologies operating at different layers provide protection and restoration capabilities at different time scales, different bandwidth granularities, and at different QoS granularities. It is a challenging task to combine in a coordinated manner the different restoration capabilities available across the layers to ensure that certain network survivability goals are met for the different services supported by the network.



A default coordination mechanism for inter-layer interaction could be the use of nested timers and current SDH/SONET fault monitoring [26]. When lower-layer recovery happens in a longer time period than higher-layer recovery, a hold-off timer is utilized to avoid contention between the different single-layer survivability schemes. Setting such timers involves a tradeoff between rapid recovery and the creation of a race condition where multiple layers are responding to the same fault, potentially allocating resources in an inefficient manner.

In other configurations where the lower layer does not have a restoration capability, there must be a mechanism for the lower layer to trigger the higher layer to take recovery actions immediately. Furthermore, faults at higher layers should not trigger restoration or protection actions at lower layers [25, 44].

### 2.3 Restoration Network Design

The *restoration network design problem* is the problem of determining the most cost-effective placement of spare capacity in the network to restore the service disrupted by any link or node failure. Several heuristics and optimization methods have been developed to solve this network design problem. In this section we briefly mention some of these approaches. A framework for *Network Engineering* which establishes capacity where it is needed by the traffic is presented in [24].

In [16] a linear programming (LP) based approach for solving path restoration problems is described. The paper is written with a specific focus on the design of the AT&T T3 restoration network, but the methodology presented can and has been used for more general network design problems. The method mainly focusses on single link failure restoration, but an LP based model for node failures can also be obtained.

The restoration network design problem is also known as the spare capacity assignment problem. In [33, pages 329–334] an optimal capacity assignment solution method, called the *square root channel capacity assignment*, is discussed. Other capacity assignment solution methods can be found in [31].

In [45] a stochastic discrete optimisation algorithm called *Simulated Allocation (SA)* is presented and compared to two other network design algorithms. The SA method can find failure disjoint (link and/or node-disjoint) working and backup paths while dimensioning the network links. Although the SA method does not guarantee optimal solutions in reasonable time, the obtained results and their comparison with the results of other methods confirm the capability of SA to find good suboptimal solutions in short computation time.



In [2, annex 5] reliable transport routing models to achieve reliable network design are presented. The basic aims of these models are to provide link diversity and protective capacity augmentation where needed so that specific network robustness objectives, such as traffic restoration level objectives, are met under failure events. This means that the network is designed so that it carries at least a fraction of traffic known as the *traffic restoration level (TRL)* under the failure event.

A set of fundamental optimisation problems for designing multi-layer telecommunication networks that are robust in the event of failures is formulated in [46]. The design problems are formulated as linear programming problems and solved through *Benders' Decomposition*, yielding an effective means for evaluating the cost of different restoration options in large B-ISDN, ATM, IP, and SDH networks.

TRee-based ACKnowledgement protocols (TRACK) use a hierarchy of dedicated servers which assures scalability to process error recovery. TRACK is designed to reliably send data from a single sender to a group of receivers and it has been recently selected as a standard-track possible architecture by the IETF, but a number of dimensioning issues remain open, such as the number of required repair servers and the optimal number of hierarchical levels. In [13] these dimensioning issues are analysed and optimal server hierarchical configurations and the optimal number of repair servers for various optimization criteria are found.

## 2.4 Traffic Engineering Recovery approaches

The main objective of MPLS Traffic Engineering (TE) is the efficient mapping of traffic demands onto the network topology to maximize resource utilization while meeting QoS constraints such as delay and packet loss. In short, the intent of TE is to put the traffic where the capacity is [4, 6]. In this section we briefly mention some TE approaches which can increase the ability of the network to quickly recovery after a failure.

### 2.4.1 Virtual Path Routing for Survivable ATM Networks

In [41], the problem of virtual path routing for survivable asynchronous transfer mode (ATM) networks is addressed. An algorithm is developed to find a virtual path configuration and bandwidth assignment that minimizes the expected amount of lost flow upon restoration from a network failure. A two-step restoration process is assumed: the first step employs a fast rerouting (self-healing) method when a failure is detected, and the second step implements a network-wide optimal reconfiguration.

The expected lost flow due to a failure on a link is computed by emulating the *k-shortest path (KSP)*-



based self-healing restoration process. In this process the entire spare bandwidth available on the shortest restoration route is used to restore as many as possible of the affected virtual paths. Then the entire spare bandwidth available on the second shortest restoration route is used to restore the remaining affected virtual paths. Thereafter the third, fourth, etc., shortest routes are used. This procedure is repeated until all the affected virtual paths are restored or until all the  $k$  possible restoration routes are exhausted.

The *survivable virtual path routing* problem can be formulated as a nonlinear, nonsmooth multi-commodity flow problem with linear constraints. A modified flow deviation method is developed to obtain a near-optimal solution, where premature convergence to a nonsmooth point could be avoided by adjusting an optimization parameter. The proposed routing scheme can detect the links that are vulnerable to a failure under the current traffic demand pattern and adjust a flow so as to improve the network survivability level.

### 2.4.2 Enhancing the self-healing capability of statically protected ATM networks

Reference [49] presents two methods of enhancing the restorability of ATM virtual path (VP) networks whereby working paths have pre-assigned protection or backup paths.

The first method is a dynamic route-searching technique which can be used when the distributed control protocol fails to activate the pre-assigned backup path(s) due to limited spare capacity caused by unforeseen multiple failures.

The second method of improving restorability involves a simple adaptation to the preplanned restoration whereby the active rather than the peak virtual path capacity is assigned to the backup path. The active VP capacity is the aggregate bandwidth of all virtual channels which are currently active on the VP and is just as accessible as the peak VP capacity.

Both methods provide a modest gain in restoration ratio where the spare capacity is very limited and instills flexibility into the restoration strategy for handling unforeseen multiple failures which affect working and backup paths.

### 2.4.3 Enhancements to Traffic Engineering for MPLS

In [42, 43] both Linear Programming (LP) and Nonlinear Programming (NLP) approaches to traffic engineering for MPLS are formulated. Four LP formulations are proposed of which two result in a large reduction in the size of the problem to be solved. The problem to be solved is the problem of

finding LSPs and assigning flows to these paths in an in some sense optimal way, given a network with link capacities and a set of traffic demands. Various notions of optimality are discussed in [42, 43]. The NLP formulation can capture effects that are important at different load regimes. A combined LP and NLP method is proposed in which the LP generates traffic paths and the NLP determines the allocation of flows to those paths.

Three enhancements to the LP/NLP algorithm, for additional capabilities to respond to different operational requirements, are presented in [14]. These enhancements are: (1) an algorithm for expanding the set of LP paths to increase the robustness of the network to uncertainties in the offered load, (2) a method of limiting the path-selection to the choice of a single explicit path for each node-pair, when such a restriction is made for reasons of administrative simplicity, and (3) an algorithm for admission control when the existing network capacity is inadequate to carry all the offered traffic.

#### 2.4.4 Protection through Thrifty Configuration

In [15] a heuristic method called *Iterative Capacity Splitting (ICS)* is proposed which finds a set of paths which can survive any single link failure. A method referred to as *Thrifty Capacity Allocation (TCA)* is proposed and applied to ICS to find a solution in which the total amount of allocated capacity is minimised.

Given a network with link capacities and traffic demands, two link-disjoint paths are found for each node-pair. The shorter of the two paths is the working (active, primary) path and the other one the protection counterpart.

The problem is formulated as a capacitated minimal cost unsplittable multi-commodity flow (MCMCF) problem which belongs to the class of NP-hard problems. (Unsplittable means that branching of flows is prohibited along the paths.)

Results found with the ICS method are compared to results found with an Integer Linear Programming (ILP) method and results found with the Simulated Allocation (SA) algorithm [45] discussed in section 2.3. Simple heuristics for improving the performance of Simulated Allocation are also presented. The TCA method can be applied to both the ILP and SA methods for reducing the capacity allocation.



### 2.4.5 Optical Network Restoration

A *generalized* version of MPLS applicable to many different network control layers, called G-MPLS (or GMPLS), has recently been proposed. A functional description of the protocol extensions needed to support GMPLS-based recovery is presented in [37].

In GMPLS, the control channel between two adjacent nodes is no longer required to use the same physical medium as the data-bearing links between those nodes. A consequence of allowing the control channel(s) between two nodes to be physically diverse from the associated data links is that the health of a control channel does not necessarily correlate to the health of the data links, and vice-versa. A link management protocol (LMP), proposed in [38], that runs between neighbouring nodes and is used to manage traffic engineering links, can be used to maintain control channel connectivity, verify the physical connectivity of the data-bearing channels, correlate the link property information, and manage link failures. LMP requires that a pair of nodes have at least one active bi-directional control channel between them.

Efforts are underway to extend the IP-based MPLS protocols to optical networks. In [19], several challenges for MPLS in optical network restoration are addressed and several enhancements for fast optical network restoration are proposed. An MPLS version applicable to the optical network called MPL(ambda)S is presented in [7].

An optical transport system multiplexes multiple optical signals onto a common fiber, necessitating the concept of a *channel*. Channel routes can be computed by means of a Constraint-based Shortest Path First (CSPF) algorithm. One of the most important criteria concerning the constrained-based path computation is the concept of the *Shared Risk Link Group (SRLG)*. An SRLG is a set of links sharing a common physical resource i.e. a common risk. By applying the SRLG constraint criteria to the constrained-based path computation, routes can be selected taking into account the resource and logical structure disjointness that implies a lower probability of simultaneous lightpath failure.

In [22], a technique to compute the SRLG with respect to a given risk type is proposed. This is achieved by identifying for a given physical layer the resources belonging to an SRLG. The proposed model also computes the dependencies of these resources on the resources belonging to lower physical layers. The result of the computation also determines the risk associated with each of the SRLGs.

### 2.4.6 Shared Backup LSP Restoration

A general concept of the sharing of links along backup paths and its requirements in terms of link state information and signalling functions is presented in [32]. In shared backup LSP restoration,



bandwidth on links on the backup path are possibly shared between backup paths of other active paths in such a way that single link, node, or Shared Risk Link Group (SRLG) failure restoration is guaranteed.

The requirements for shared backup LSP restoration are: (1) the link state protocols should convey the total bandwidth used on the link for active LSPs, the total bandwidth used on the link for backup LSPs, and the total available bandwidth on the link, and (2) the signalling protocol information elements should consist of the setup information and procedures for a backup LSP, and the association between the active and backup LSP, and explicit hop information about the active and backup LSPs at the level of the failure entity to protect against.

#### 2.4.7 Fast Reroute Restoration

The ability to quickly reroute traffic around failed links and nodes in a label switched path can be extremely important to users. Several schemes for performing fast rerouting (local repair) have been proposed to minimize the overhead of LSP restoration as well as the overhead of backup LSP computation.

In [21], a method for setting uni-directional *alternative label switched paths* to perform fast rerouting is defined. A portion of an LSP (which may include the entire primary path) that is to be protected by an alternative path is referred to as the *protected path segment*. The ingress and egress endpoints of the protected path segment are referred to as the *source* and *destination switch* respectively. The main idea behind the fast rerouting method is to reverse the traffic at the point of failure along the protected LSP back to the source switch of the protected LSP such that the traffic flow can be then redirected via a node-disjoint LSP between the source and destination switches of the protected LSP segment. This method can also provide an in-band means for the quick detection of link and switch failures or congestion along a primary path without resorting to an out-of-band signalling mechanism. As soon as a switch along the primary path (on the protected segment) detects a traffic flow on the alternative path segment that runs in the reverse direction of the primary path, it may stop sending traffic downstream of the primary path (along the protected segment) by initiating an immediate rerouting of data traffic to the alternative path at the source switch.

In [29], *crankback routing* extensions for CR-LDP signalling and for RSVP-TE signalling are proposed which can be applied to LSP restoration. Crankback routing requires notifying an upstream LSR of the location of the blocked (or failed) link or node. CR-LDP (Constraint-based Routing Label Distribution Protocol) and RSVP-TE (RSVP Extensions for LSP Tunnels) can be used for establishing explicitly routed LSPs in an MPLS network. Explicit paths can be designated based on the distributed



information at the LSR initiating an LSP. The information available at the initiating LSR may be out of date. This may lead to a blocked LSP setup request due to insufficient resources along the selected path and may result in the LSP setup being abandoned. If the ingress or intermediate area border LSR knows the location of the blocked (or failed) link or node, the LSR can designate an alternate path and then reissue the setup request, which can be achieved by a mechanism known as crankback routing.

In [28], a fast reroute method for automatically setting up detour paths over a RSVP signaled LSP is explained. The backup or detour LSPs originate from  $(N - 1)$  nodes along the primary path and are node-disjoint from the primary path, where  $N$  is the number of hops that the LSP traverses. The backup LSPs are merged with the primary LSP to make as short a path as possible to minimize the overhead involved in LSP computation.

The fast reroute method [28] mentioned in the previous paragraph and the shared backup LSP restoration principles [32] are used to find *shared fast reroute LSPs*. In [30] new signalling procedures for RSVP-TE signalling that allow the implementation of *shared fast reroute LSPs* are outlined. The proposed procedures are used to compute and establish the shared fast reroute LSPs in a distributed fashion, and are used to continuously adapt to the latest topology without manual intervention. The method is also extended to support multiple load-balanced primary LSPs that have shared backup LSPs.

## Chapter 3

# Linear and Nonlinear Programming Solutions for MPLS Routing

The MPLS routing problem is the problem of finding an in some sense optimal set of label switched paths (LSPs) and assigning flow to these paths such that no capacity constraints are violated. Given a network with link capacities and a set of traffic demands, we want to find traffic engineered LSPs to carry the offered traffic in an in some sense optimal way. Various notions of optimality exist. In this chapter we find a routing solution which will maximize the spare capacity (or slack) of the network. The more slack available in a network, the greater the probability to quickly find another stable, optimal routing solution after a link failure detection.

The remainder of this chapter is organized as follows. Section 3.1 presents a linear programming method and section 3.2 presents a nonlinear programming method for solving the MPLS routing problem. The results of the two optimization problems are presented in section 3.3. We compare in particular the distribution of the link utilizations as computed by these two methods and show that for the network models tested, parameter values can be chosen so that both methods give similar link utilization results. We also compare the two sets of active (working) paths found by the linear and nonlinear optimization methods. The conclusions are stated in section 3.4.

### 3.1 A Linear Optimization Method

Consider a communications network consisting of  $N$  nodes and  $L$  links. Let  $\mathcal{N} = \{1, 2, \dots, N\}$  denote the set of nodes and let  $\mathcal{L} = \{1, 2, \dots, L\}$  denote the set of links. The nodes represent the routers in the MPLS-capable part of a network. Some nodes are connected by a link. The links are directed: each link has a starting node and an ending node which are routers from the set  $\mathcal{N}$ .



Let  $d_{(m,n)}$  denote the predicted demand (offered load) of traffic that wants to enter the MPLS network at node  $m$  and wants to exit at node  $n$ . We assume that the demands  $d_{(m,n)}$  and the link capacities  $b_{(m,n)}$  are such that a feasible solution exists. The definition of feasibility will be given shortly.

A path  $P$  is a sequence of links. In our terminology a route and a path and an LSP (label switched path) are synonymous. No path traverses the same link or the same node more than once. Any assignment of the demands  $d_{(m,n)}$  to paths in the network leads to link loads  $X_{(m,n)}$  where  $X_{(m,n)}$  is the load on link  $(m, n)$ . Let  $x_{(i,j)(m,n)}$  denote the flow of the traffic of node-pair  $(i, j)$  on link  $(m, n)$ .

For some purposes it is more convenient to work with the slacks  $S_{(m,n)}$  defined by

$$S_{(m,n)} = b_{(m,n)} - X_{(m,n)} \quad (3.1)$$

We call a routing problem feasible if there exists a solution with  $S_{(m,n)} > 0$  for all  $(m, n)$  with  $b_{(m,n)} > 0$ .

### 3.1.1 Optimization Criteria

The general objective is to minimize the weighted sum of the link loads  $X_{(m,n)}$ . Thus we begin with the weakest optimality criterion:

*Definition:* A routing scheme is *non-dominated* if it results in link loads  $X_{(m,n)}^*$  with the property that there does not exist another routing scheme with link loads  $X_{(m,n)}$  with

$$X_{(m,n)} \leq X_{(m,n)}^* \text{ for all } (m, n), \text{ and } X_{(m,n)} < X_{(m,n)}^* \text{ for at least one } (m, n).$$

In [43] four different optimization criteria are discussed. The next section focusses on the third criterion: a combined parametric criterion.

### A Combined Parametric Criterion

We consider the following objective, parametrized by  $\varepsilon \geq 0$ :

$$\text{Maximize} \left[ \varepsilon Z + \sum_{(m,n)} w_{(m,n)} S_{(m,n)} \right] \quad (3.2)$$

or equivalently, by substituting (3.1) into (3.2):

$$\text{Maximize} \left[ \varepsilon Z + \sum_{(m,n)} w_{(m,n)} [b_{(m,n)} - X_{(m,n)}] \right]$$

or equivalently,

$$\text{Minimize} \left[ -\varepsilon Z + \sum_{(m,n)} w_{(m,n)} X_{(m,n)} \right] \quad (3.3)$$

subject to the following constraints:

$$x_{(i,j)(m,n)} \geq 0; Z \geq 0 \quad (3.4)$$

$$\sum_{(i,j)} x_{(i,j)(m,n)} + C_{(m,n)} Z \leq b_{(m,n)} \quad (3.5)$$

for each  $(m, n)$  with  $b_{(m,n)} > 0$ , and subject to the conservation-of-flow constraint:

$$\delta_{i,n} d_{(i,j)} + \sum_k x_{(i,j)(k,n)} = \delta_{n,j} d_{(i,j)} + \sum_k x_{(i,j)(n,k)} \quad (3.6)$$

at each node  $n$  where  $\delta_{i,n}$  equals 1 if  $i = n$ , and 0 otherwise.

In the special case where we choose all the link weights equal  $w_{(m,n)} \equiv w > 0$  independent of link  $(m, n)$ , we are in effect minimizing the average hop-count of the traffic, subject to the feasibility constraints. In another special case, if we make the weight  $w_{(m,n)}$  equal to the propagation delay of link  $(m, n)$ , we are minimizing the sum of the propagation delays, which was the intention of the original form of OSPF (Open Shortest Path First) routing. In general, we can think of the weights  $w_{(m,n)} > 0$  as the OSPF costs of the links.

The coefficients  $C_{(m,n)} > 0$  are either given or chosen in some manner. The choice of  $C_{(m,n)}$  is of secondary importance. It is stated in [43] that if one takes  $C_{(m,n)} = b_{(m,n)}$ , one maximizes the normalized link slack (or minimizes the maximum link utilization) in the network. When there is a feasible solution to the routing problem, it could be argued that the choice  $C_{(m,n)} = \sqrt{b_{(m,n)}}$  has the effect of minimizing the maximal probability of a link flow exceeding its bandwidth.

For any  $\varepsilon \geq 0$ , an optimum solution is non-dominated.



### 3.1.2 Linear Program (LP) Solutions

Four linear program formulations based on the above criteria are investigated in [43]. In the pair-based flow formulation, the traffic of each node pair is treated as a separate commodity. In the egress-centric flow formulation the total traffic exiting the network at each node (from all originating nodes) is defined as a commodity. The ingress-centric flow formulation is analogous to the egress-centric case, with the total traffic entering the network at each node (from all originating nodes) defined as a commodity. In the fourth formulation, the path-based formulation, a set of admissible paths between each node pair is defined as a commodity. The egress- and ingress-centric formulations lead to linear programs with fewer variables than the pair-based formulation. The path-based formulation has the least number of variables but the solution method is iterative and may require resolving the problem a large number of times. The egress-centric flow formulation is discussed next.

#### Egress-centric Flow Formulation

In this LP formulation, the total traffic exiting the network at each node (from all originating nodes) is defined as a commodity. Thus, we define

$$D_j^{(out)} = \sum_i d_{(i,j)} \quad (3.7)$$

We define  $x_{(m,n),j}$  as the amount of flow carried on link  $(m,n)$  that is destined for node  $j$ . Thus, we have the LP for the egress-centric formulation:

$$\text{Minimize} \left[ -\varepsilon Z + \sum_{(m,n)} w_{(m,n)} \sum_j x_{(m,n),j} \right] \quad (3.8)$$

subject to the constraints:

$$x_{(m,n),j} \geq 0; Z \geq 0 \quad (3.9)$$

$$\sum_j x_{(m,n),j} + C_{(m,n)} Z \leq b_{(m,n)} \quad (3.10)$$

for each  $(m,n)$  with  $b_{(m,n)} > 0$ , and subject to the conservation-of-flow constraint:

$$d_{(n,j)} + \sum_k x_{(k,n),j} = \delta_{n,j} D_j^{(out)} + \sum_k x_{(n,k),j} \quad (3.11)$$

at each node  $n$ , for each destination  $j$ .

The egress-centric LP has  $(NL + 1)$  variables,  $N^2$  equality constraints, and  $L$  inequality constraints. For each commodity, the equality constraints sum to zero and we have a total of  $N(N - 1)$  linearly independent equality constraints.

### Determining Paths from the LP Solution

For the egress-centric formulation, for each demand destination  $j$ , we have a flow using the variables  $x_{(m,n),j}$  for the various links  $(m, n)$ , where the offered load at each node  $i$  is  $d_{(i,j)}$ . We want to decompose this flow into a sum of flows on paths terminating at  $j$ .

For a destination node  $j$ , consider all links  $(m, n)$  with  $x_{(m,n),j} > 0$ . Repeat the following steps until  $x_{(m,n),j} = 0$  for every link  $(m, n)$ .

1. Find a node  $i$  with the following two properties: (1)  $x_{(i,n),j} > 0$  for at least one node  $n$  and (2)  $x_{(m,i),j} = 0$  for all nodes  $m$ .
2. Use a path-finding method to find an  $i \rightarrow j$  path  $P$ . Different path-finding methods may influence, for example, how the paths for a single destination vary in length, but it suffices to perform a directed random walk starting at node  $i$ .
3. Let  $\ell_P$  be the minimum of  $x_{(m,n),j}$  over all links  $(m, n)$  in  $P$ . Assign a flow of  $\ell_P$  to  $P$  and subtract  $\ell_P$  from  $x_{(m,n),j}$  for all  $(m, n)$  in  $P$ . At least one link has had its flow reduced to zero for this demand pair.

Each path found corresponds to at least one positive component of  $x$ . As stated above, the LP has  $N(N - 1) + L$  linearly independent constraints, so if  $x$  and  $Z$  provide a basic solution to the LP and  $Z$  is positive,  $x$  will have at most  $N(N - 1) + L - 1$  positive components. Thus we have at most  $N(N - 1) + L - 1$  paths.

The set of paths computed from the LP solution is not unique. The sequence of  $i$  nodes chosen in step (1), as well as the methods used to find the  $i \rightarrow j$  path influence which paths are chosen. One would like to choose the set of paths in an “in some sense optimal” way. The next section provides a few strategies that can be used to find different sets of paths from the LP solution.



**Different strategies to use when choosing a set of paths**

In step (1) above there may exist two or more different nodes which could be chosen as the valid node  $i$ . Any one of these nodes may be randomly selected, or, if specific requirements on the set of paths are desirable, these should be taken into consideration. For example, suppose either node  $k$  or node  $\ell$  were valid candidates for node  $i$ , but  $k \rightarrow j$  paths which use specific links (or paths which meet some other criteria) are preferred. In this case, one should first select node  $k$  as the valid node  $i$  in step (1). Then, in step (2), one would try to identify paths which better suit the preferred criteria.

In step (2), there may exist several possible  $i \rightarrow j$  paths. One may prefer to choose a path according to (a) the path length (shortest/longest), or (b) the links included/excluded in the path, or (c) the amount of traffic which could be assigned to the path, or (d) other criteria.

One may decide that the “in some sense optimal” way of choosing a set of paths would be to find for each source-destination (S-D) pair as many link-disjoint multipaths as possible. Then, should a link fail, an alternative path for each S-D pair exists. In this case, one should find the set of all possible  $i \rightarrow j$  paths in step (2) and decide which combination of paths (if any) will best satisfy this criteria.

If the set of all possible  $i \rightarrow j$  paths only consists of two paths  $P_1$  and  $P_2$  which share one or more links and  $\max(\ell_{P_1}, \ell_{P_2}) = d_{(i,j)}$  then if one prefers multipaths one may want to split the flow  $d_{(i,j)}$  over two paths in stead of assigning all the flow to one path.

**3.2 A Nonlinear Optimization Method**

In this section we use a criterion that has a nonlinear dependence on link flows in order to keep the slacks on links bounded away from zero. The advantage of using this nonlinear criterion over a linear criterion is that the nonlinear criterion can reflect the different desired effects that are important at different load regimes (see section 3.2.3).

The Flow Deviation (FD) Algorithm [9, 31, 33] can be used to solve constrained nonlinear optimization problems if the objective function is convex. The convexity of the objective function guarantees that any local minimum found is also a global minimum. Another important requirement for the FD algorithm is that the objective function should be separable with respect to the links so that the objective function is a sum of functions defined on the links. If both these conditions hold, the flow deviation algorithm can be used to solve the optimization problem.

The original flow deviation algorithm is described in Kleinrock’s book [33]. The Bertsekas-Gallager flow deviation algorithm is described in [9], and another FD version with a simple method of finding



an initial feasible solution is described in Kershenbaum's book [31]. The original and derived versions of the flow deviation algorithm are considered in [8, 18]. An approximate implementation yielding near optimal results is described in [12]. The mathematical methods on which the flow deviation algorithms are based are also reviewed in [18].

In this section we formulate the problem as the *LSP design problem* in which we find a set of LSPs and assign optimal bandwidths to these paths such that all the offered traffic can be carried.

### 3.2.1 Feasibility and Optimality

Each path  $P$  will be assigned a bandwidth  $B_P \geq 0$ . The goal is to select these bandwidths in an in some sense optimal way. Let  $\mathcal{P}$  denote the set of all paths. Let  $\mathcal{P}^{(i)}$  denote the set of paths that utilize link  $i$ . Let  $\mathcal{P}_{(m,n)}$  denote the set of paths from node  $m$  to node  $n$  with  $m \neq n$ . Let  $\mathbf{B} = (B_P)_{P \in \mathcal{P}}$  denote a set of bandwidths.  $\mathbf{B}$  is said to be feasible if the following two constraints hold:

1. For each node-pair  $(m, n)$ :  $\sum_{P \in \mathcal{P}_{(m,n)}} B_P = d_{(m,n)}$  so that all of the offered traffic is carried.
2. For each link  $i$ :  $\sum_{P \in \mathcal{P}^{(i)}} B_P \leq b_i$  so that no link has an offered load greater than its capacity.

We next choose a definition of optimality. Let  $f_i = \sum_{P \in \mathcal{P}^{(i)}} B_P$  denote the flow on link  $i$ . Let  $f_i/b_i$  denote the utilization of link  $i$  and let  $s_i = b_i - f_i$  denote the slack on link  $i$ .

Let  $F_i(f_i)$  denote a penalty function for link  $i$  when the link carries a flow  $f_i$ . The *LSP design problem* is specified in terms of the following constrained nonlinear optimization problem: Find a set of feasible bandwidths  $\mathbf{B}_{\text{opt}}$  that minimizes the objective function

$$F(\mathbf{B}) = \sum_i F_i(f_i) \quad (3.12)$$

subject to the two constraints above where the sum in equation (3.12) is over links  $i$  with  $b_i > 0$ .  $\mathbf{B}_{\text{opt}}$  is said to provide an optimal solution to equation (3.12). Note that the optimal link flows  $f_i$  are unique although the optimal bandwidths  $\mathbf{B}$  are usually not: this matter is discussed in appendix B.

### 3.2.2 The Penalty Function

The link penalty functions  $F_i(x)$  used in the LSP design problem have at least three roles. First, they must to a reasonable degree represent an intuition of what constitutes a “good” load balancing scheme. Second, they must be an efficient way of managing constraints, in particular the constraint that no link



carries a load larger than, or even close to, its bandwidth. Third, the penalty functions must make it possible to efficiently find an optimal solution to the LSP design problem.

The flow deviation algorithm requires that the penalty functions  $F_i(x)$  be increasing and convex on  $[0, b_i)$  with  $\lim_{x \uparrow b_i} F_i(x) = +\infty$ . The latter requirement necessitates a minor change to the definition of feasibility: a solution is said to be feasible if  $f_i < b_i$  (strict inequality) on all links  $i$ . It is also convenient to make a slightly stronger demand on the functions  $F_i(x)$  and require that each  $F_i(x)$  be “strongly monotone” on the interval  $[0, b_i)$  so that the penalty functions are non-negative on  $[0, b_i)$  and their derivatives with respect to flow are positive on  $(0, b_i)$ .

Previous studies of the flow deviation algorithm [9, 31, 33] used

$$F_i(x) = M_i \frac{x}{b_i - x} \quad (3.13)$$

as a link penalty function. If we assume that the offered load to each link  $i$  is a Poisson process of packet arrivals and that packets have independent, identically distributed sizes with exponential distribution and average  $M_i$ , and that there is an infinite buffer, and that the resulting utilization of the link is  $x/b_i$ , then the penalty function (3.13) is the product of the flow  $x$  and the average delay (waiting and service both included, but propagation delay excluded). With the M/M/1 assumptions above, the sum of the link penalty functions is a measure of the average total network delay.

However, in the modern Internet with TCP, and Random Early Detection (RED) and all its variations, it is possible to have very highly utilized links (utilization practically one) and still low delay and low loss in the buffer: all delay is moved to the edge of the network. The same holds for example for ATM with Available Bit Rate (ABR), in particular the Explicit Rate (ER) version of ABR. Equation (3.13) is probably no longer a suitable penalty function. Given these concerns, we present a link penalty function with properties which make it suitable for use in an objective function whose minimization will yield routes and bandwidths that correspond closely to the optimal operation of a modern internet. Our choice of penalty function is

$$F_i(x) = c_i x + \eta \sigma_i \left( \frac{\sigma_i}{b_i - x} \right)^\nu \quad (3.14)$$

where link  $i$  has a bandwidth  $b_i \geq 0$ , a weight factor  $\sigma_i > 0$  with  $\eta > 0$ ,  $\nu > 1$  and  $F_i(x) = \infty$  if  $x \geq b_i$ . The factor  $c_i$  is explained below. The function (3.14) is strongly monotone and the first derivative of the penalty function is

$$F'_i(x) = \frac{d}{dx} F_i(x) = c_i + \eta \nu \left( \frac{\sigma_i}{b_i - x} \right)^{\nu+1}. \quad (3.15)$$

Let  $\tau_i \geq 0$  denote the propagation delay on link  $i$ . Set  $F'_i(0) = \tau_i$ . Then  $c_i = \tau_i - \eta \nu (\sigma_i/b_i)^{\nu+1}$ . The properties of the link penalty function (3.14) under light and heavy load are discussed in the following section.



### 3.2.3 Behavior under Light and Heavy Load

With reference to the link penalty function (3.14) we choose  $\eta$  positive but small so that if a feasible solution exists for which all flows  $f_i$  are small and all link utilizations  $f_i/b_i$  are low – in which case the system is said to be uniformly lightly loaded – then the penalty function (3.14) will yield routes that are in agreement with OSPF routing where the propagation delays are the OSPF metrics of the links.

If the system is not uniformly lightly loaded then the penalty function enforces a distance from the barrier  $b_i$ . The parameter  $\eta$  determines when the barrier  $b_i$  begins to dominate the initial linear behaviour of the penalty function. A larger value of  $\eta$  causes the penalty function to rise earlier when the flow approaches the barrier. The parameter  $\nu$  determines the behaviour of the penalty function as it approaches the barrier  $b_i$ . A larger value of  $\nu$  makes the penalty function steeper when the flow approaches the barrier.

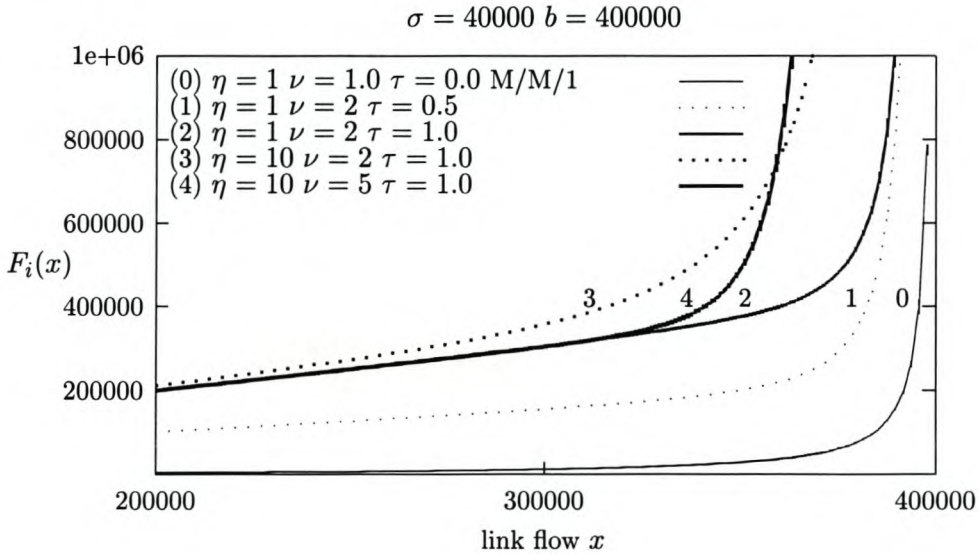


Figure 3.1: Examples of the penalty function

Figure 3.1 plots the penalty function (3.14) of a link  $i$  as a function of the link flow  $x$ . The link bandwidth  $b_i = 400,000$  and the weight  $\sigma_i = b_i/10 = 40,000$ . These values are related to the parameters of a 50-node network model [50] where the average link capacity is  $190,689 \pm 81,026$  and the average flow carried on a link is  $95,265 \pm 48,414$ .

With reference to Fig. 3.1 plot (0) shows the M/M/1 penalty function using related parameters. Plots (1) through (4) are for the penalty function (3.14). Plot (1) shows the effect of  $\tau_i = 0.5$ ,  $\eta = 1$

and  $\nu = 2$ . Plot (2) shows the effect of increasing  $\tau_i$  from 0.5 to 1.0. The parameter  $\eta$  determines when the barrier  $b_i$  begins to dominate the initial linear behaviour of the penalty function. Plot (3) shows that the penalty function begins to rise towards the barrier earlier when  $\eta$  is increased from 1 to 10. The parameter  $\nu$  determines the behaviour of the penalty function as it approaches the barrier  $b_i$ : increasing  $\nu$  increases the steepness of the rise. Plot (4) shows the effect of increasing  $\nu$  from 2 to 5.

### 3.2.4 The Flow Deviation Algorithm

The operation of the flow deviation algorithm is simple. The algorithm executes in a loop where each iteration of the loop implements one step of the algorithm. During each step the algorithm computes the current set of shortest (least cost) paths from all sources to all destinations. An optimal amount of flow is diverted from the current set of LSPs to the shortest paths. Those shortest paths that are not already in the LSP set are added to the LSP set, the link costs are updated (the link costs have changed because the link flows have changed) and the next step of the algorithm is executed. The loop continues until flow re-distribution achieves no further reduction in the objective function. A small worked example of the operation of the flow deviation algorithm can be found in [31].

#### The Algorithm

In the MPLS context the flow deviation algorithm incrementally improves the set  $\mathcal{P}$  of LSPs and improves the distribution of traffic over multiple paths in  $\mathcal{P}$  from the same source to the same destination. Improving  $\mathcal{P}$  mainly consists of adding paths that have, or are likely to have, lower cost than the existing paths from the same source to the same destination. Improving  $\mathcal{P}$  may involve discarding paths  $P$  that are known not to have positive  $B_P$  in any optimal solution, or are not likely to have such a positive flow. Discarding non-promising paths is not necessary for convergence but significantly decreases the computational effort.

The algorithm executes in a loop. Each iteration of the loop implements one step which is identified by a step index  $k$ .

1. Initialize: Set  $k = 0$ . For each link  $i$  set the link flow  $f_i = 0$ . Compute the least cost path  $P = P_e(m, n)$  connecting each node pair  $(m, n)$ . Set the target bandwidth  $B_P = d_{(m, n)}$ . If necessary call step (6) to enforce a feasible solution. Initialize the path set  $\mathcal{P} = \cup_{(m, n)} P_e(m, n)$ .
2. For each link  $i$  compute the link cost  $C_i^L = F'_i(f_i)$ . For each path  $P$  compute the route cost  $C_P^R = \sum_{i \in \mathcal{L}_P} C_i^L$ .



3. Compute a feasible direction  $\Delta = (\Delta_P)_{P \in \mathcal{P}}$  and an improved path set  $\mathcal{P}$ . The calculation of an improved path set and a feasible direction is discussed in the next section.
4. Convergence test: If no feasible direction can be found then optimality has been achieved and the algorithm halts. This stopping rule is theoretically correct but of no practical value. A practical stopping rule is discussed on page 29.
5. Compute improved path flows  $\mathbf{B}$ : Compute a value of  $x$  such that  $B_P := B_P + x\Delta_P$  yields a value  $F(\mathbf{B})$  of the objective function which is a strict improvement over the value of the objective function computed in the previous step, and in the direction  $\Delta$  is optimal. This computation is called the “line search” for  $x$ . The calculation of  $x$  is discussed on page 28. In qualitative terms: a very small positive  $x$  value always gives an improvement. We increase  $x$  either until the objective function stops decreasing, or until a path flow  $B_P$  goes to zero in which case the path  $P$  leaves the set  $\mathcal{P}$ .
6. Enforce a feasible solution: If the target bandwidths  $\mathbf{B}$  are not feasible then for each link  $i$  set  $b_i := \alpha b_i$  where  $\alpha = \max_i(1.05f_i/b_i)$ . The solution  $\mathbf{B}$  is now feasible.<sup>1</sup>
7. Compute improved link flows: For each link  $i$  compute  $f_i := f_i + x\delta_i$  where  $\delta_i := \sum_{P \in \mathcal{P}(i)} \Delta_P$ .
8. Loop statement:  $k := k + 1$  and go to step 2.

### Choosing a Feasible Direction

A feasible direction is a map  $\Delta = (\Delta_P)_{P \in \mathcal{P}}$  with the following properties:

- the traffic demand offered to each node pair  $(m, n)$  is constant therefore  $\sum_{P \in \mathcal{P}(m, n)} \Delta_P = 0$ ,
- an empty path cannot have its bandwidth allocation lowered so that if  $B_P = 0$  then  $\Delta_P \geq 0$ ,
- a feasible direction will lower the network cost so that  $\sum_{P \in \mathcal{P}} \Delta_P C_P^R < 0$ .

Two methods for computing a feasible direction are presented and compared in [11]. The first method, the so-called global method, may add paths to the set  $\mathcal{P}$ . The second method, the so-called local method, does not add paths to the set  $\mathcal{P}$ : in fact it is likely to remove paths from  $\mathcal{P}$ . In this thesis we have used only the global method which is explained next. See [11] for an explanation of the local method.

---

<sup>1</sup>When the flow deviation algorithm terminates then  $\alpha = 1$  else the solution  $\mathbf{B}$  is not feasible.

Given a feasible solution  $\mathbf{B}$  and the current link costs  $C_i^L$ , compute the shortest path  $P_e(m, n)$  connecting each source-destination pair  $(m, n)$ . There may be several such paths in which case a tie-breaking mechanism is needed. This path may already be in the set of known paths  $\mathcal{P}_{(m,n)}$  and have a positive flow  $B_{P_e(m,n)} > 0$ . If the path is not in  $\mathcal{P}_{(m,n)}$  then it is added to  $\mathcal{P}_{(m,n)}$ . For each  $P \in \mathcal{P}_{(m,n)}$  compute

$$\Delta_P = \begin{cases} -B_P & P \in \mathcal{P}_{(m,n)} \setminus P_e(m, n) \\ d_{(m,n)} - B_P & P = P_e(m, n). \end{cases}$$

### The Line Search

In this section we compute a value of  $x$  which yields an improved solution

$$B_P(x) = B_P + x\Delta_P \quad (3.16)$$

for all  $P \in \mathcal{P}$ . Define

$$x_{max}^L = \min_{i: \delta_i > 0} \frac{s_i}{\delta_i}$$

where  $x_{max}^L = +\infty$  if  $\delta_i \leq 0$  for all  $i$ . If  $x$  grows to  $x_{max}^L < \infty$  then the slack on one or more links will equal zero. Thus we have the constraint  $x < x_{max}^L$ . Next define

$$x_{max}^R = \min_{P: \Delta_P < 0} \frac{B_P}{|\Delta_P|}.$$

It is impossible that  $\Delta_P \geq 0$  for all  $P$ . If  $x$  grows to  $x_{max}^R$  then the flows on one or more routes in  $\mathcal{P}$  will decrease to zero. Thus we have the constraint  $x \leq x_{max}^R$ .

Set  $x_{max} = \min(x_{max}^L, x_{max}^R)$ . With an abuse of notation, we wish to find a value of  $x \in [0, x_{max}]$  which minimizes

$$F(x) = \sum_i F_i(f_i + x\delta_i).$$

Setting  $y = f_i + x\delta_i$  and taking derivatives we obtain

$$F^{(k)}(x) = \left(\frac{d}{dx}\right)^{(k)} F(x) = \sum_i (\delta_i)^k \left(\frac{d}{dy}\right)^{(k)} F_i(y).$$

Since the functions  $F_i(\cdot)$  are strongly monotone, even derivatives of  $F(x)$  are positive and odd derivatives of  $F(x)$  are strictly increasing. If

$$x_{max}^R < x_{max}^L \quad (3.17)$$



then  $x_{max} = x_{max}^R$ . In that case, compute

$$F'(x_{max}) = F^{(1)}(x_{max}) \quad (3.18)$$

If equation (3.17) holds and  $F'(x_{max}) \leq 0$  then  $x_{max}$  is the optimal value for  $x$ . In this case, in updating the feasible solution, one or more routes have their flow reduced to zero and these routes may be removed from  $\mathcal{P}$ .

If equation (3.17) does not hold, or if it holds but  $F'(x_{max}) > 0$  then we need to find the value of  $x \in [0, x_{max})$  where  $F'(x) = 0$ . Because the even derivatives of  $F(x)$  are positive we can use the Newton-Raphson method [47] to find the value of  $x$ .

### The Stopping Rule

The algorithm requires a stopping rule to determine the iteration  $k$  when the algorithm has converged. We can stop when either  $|F'(x_{k+1}) - F'(x_k)|$  or  $|x_{k+1} - x_k|$  has been close to zero for some time in which case further iterations will yield no improvement in the solution. We can use a combination of these two criteria. Because even derivatives of  $F(\cdot)$  are positive, the sequence  $(x_k)$  will become monotone decreasing or increasing. It may be safe not to stop until the sequence has been monotone for some time *and* one or both of the other conditions above is satisfied.

### Implementation Issues

Each time the global method is invoked it calculates the shortest paths connecting all source-destination pairs and checks whether the current set of shortest paths is already in  $\mathcal{P}$ . These calculations are computationally expensive. The shortest paths are computed using Floyd's algorithm – see [3] and the references therein for a discussion of the relative merits of several well-known shortest path algorithms. Each path  $P$  is stored in a table which is accessed via a hash index computed over the link set  $\mathcal{L}_P$ .

### 3.3 Experimental results

This section presents and compares the linear and nonlinear optimization results for the 10-node, the 20-node and the 100-node network models described in appendix A.

The linear objective function (3.8) used the value  $\varepsilon = 0.05$ , constraint (3.10) used  $C'_{(m,n)} = b_{(m,n)}$  and the linear programming solver, lp\_solve [40], was used to solve the Egress-centric Flow Formulation. Lp\_solve solved the 100-node network (with load factor = 10.00) in  $\pm 6$  minutes on a Pentium-III 500 MHz machine. The paths were determined with a random walk greedy algorithm.

The nonlinear penalty functions (3.14) used the values  $\tau_i = 1$  and  $\sigma_i = \max(1.0, b_i/10.0)$ .

#### 3.3.1 Link utilization results from the LP solution and the Flow Deviation program

Figures 3.2, 3.3 and 3.4 represent the link utilization results from the LP and Flow Deviation solutions for the 10-node, the 20-node and the 100-node models respectively. The link utilization results for the 50-node model are not included because lp\_solve could only solve the model with a load factor of 0.20 which results in link utilizations  $\leq 0.30$ .

The top bar charts on pages 32–34 compare the Flow Deviation utilization results with the lp\_solve utilization results. The LP maximizes the normalized link slack (or minimizes the maximum link utilization). Different  $\eta$  and  $\nu$  values were used for the Flow Deviation program in a search for a combination of  $\eta$  and  $\nu$  values which force the Flow Deviation program to have more or less the same utilization results as the LP solution.

The following  $\eta$  and  $\nu$  combinations give Flow Deviation utilization results which are more or less the same as the lp\_solve utilization results: for the 10-node model  $\eta = 1$  and  $\nu = 40$ , for the 20-node model  $\eta = 20$  and  $\nu = 2$ , and for the 100-node model  $\eta = 1$  and  $\nu = 2$ .

The top bar charts of figures 3.2 and 3.3 show that the number of links with utilization between 40 and 50 percent differ by 10 links and 16 links for the 10-node and the 20-node networks respectively. Although this might seem like a significant difference, we consider the Flow Deviation utilization results as more or less the same as the LP utilization results because the *important* link utilization categories (the link utilizations greater than 50 percent) do not differ that much.

The middle and bottom bar charts on pages 32–34 display the link utilization information of the Flow Deviation results using different  $\nu$  and  $\eta$  values. In the middle bar chart the  $\eta$  value is constant and in the bottom bar chart the  $\nu$  value is constant.



### 3.3.2 Paths derived from the LP solution compared to paths found with the Flow Deviation program

In the linear objective function (3.8), all the weights  $w_{(m,n)} = 1$ . This has the effect of minimizing the average hop-count of the traffic. Both a random walk and a shortest path path-extracting method (see section 3.1.2) were implemented and for the four models solved, both these methods compute almost the same optimal set of paths.

The pie chart figures 3.5–3.7 compare the set of random walk paths found from the LP solution with the set of paths found with the Flow Deviation (FD) program.

Let  $r_1$  and  $r_2$  denote the same route (or path), where  $r_1$  belongs to the FD path set and  $r_2$  to the LP set of paths. Let  $f_1$  and  $f_2$  denote the flow carried by  $r_1$  and  $r_2$  respectively.

The pie charts on the left of figures 3.5–3.7 compare the routes. If

$$\frac{|f_1 - f_2|}{\max(f_1, f_2)} < 0.05$$

then the route is counted under the “strong agreement” pie chart slice, else the route is counted under the “weak agreement” slice.

The “Only in FD” and “Only in LP” slices are self explanatory.

The pie charts on the right of figures 3.5–3.7 compare the sum of the flows carried by the routes. If

$$\frac{|f_1 - f_2|}{\max(f_1, f_2)} < 0.05$$

then the route flow  $\max(f_1, f_2)$  is counted under the “strong agreement” pie chart slice, else  $\max(f_1, f_2)$  is counted under the “weak agreement” slice.

The pie charts reveal that the Flow Deviation method finds many more paths than the random walk LP path-finding method. One reason for this may be that Flow Deviation prefers choosing multipaths (see [8, 11] or appendix B) whereas the random walk path-finding method will choose only one path randomly and assign all the flow to it if possible.

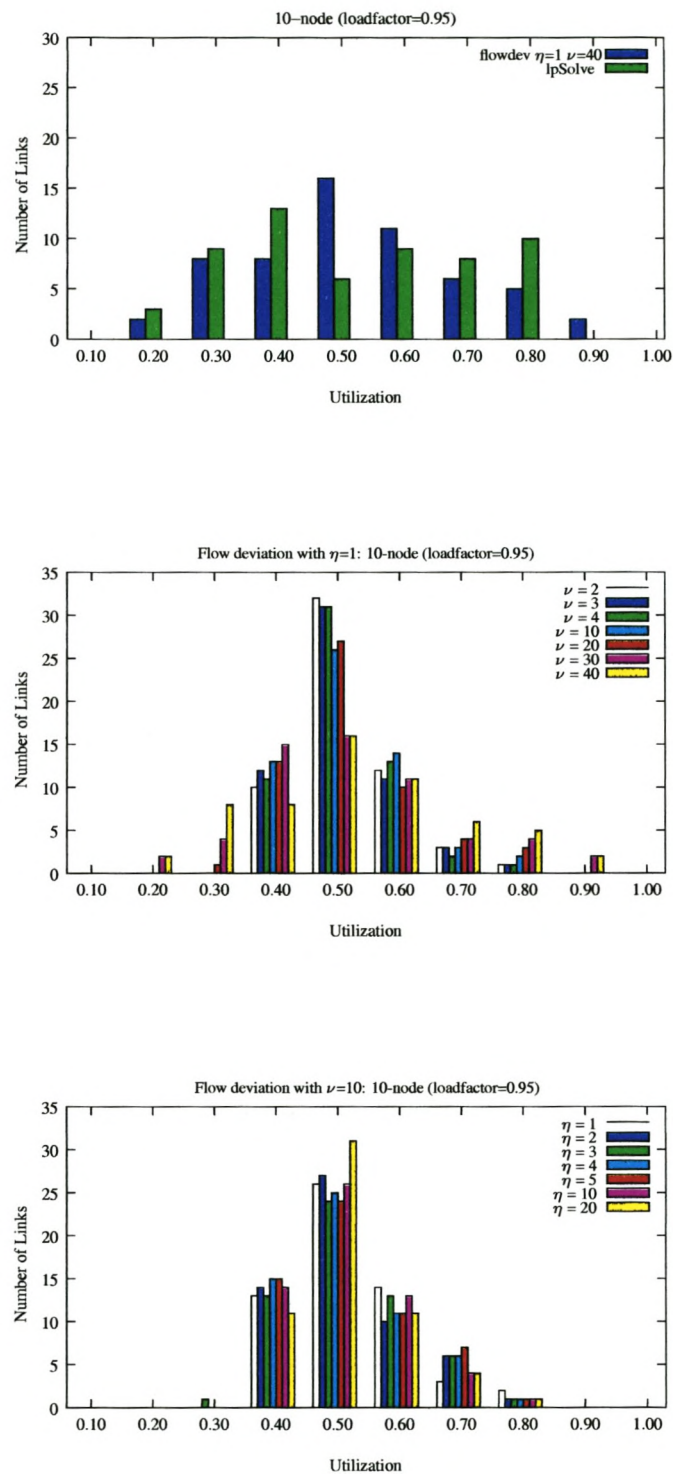


Figure 3.2: 10-node utilization bar charts



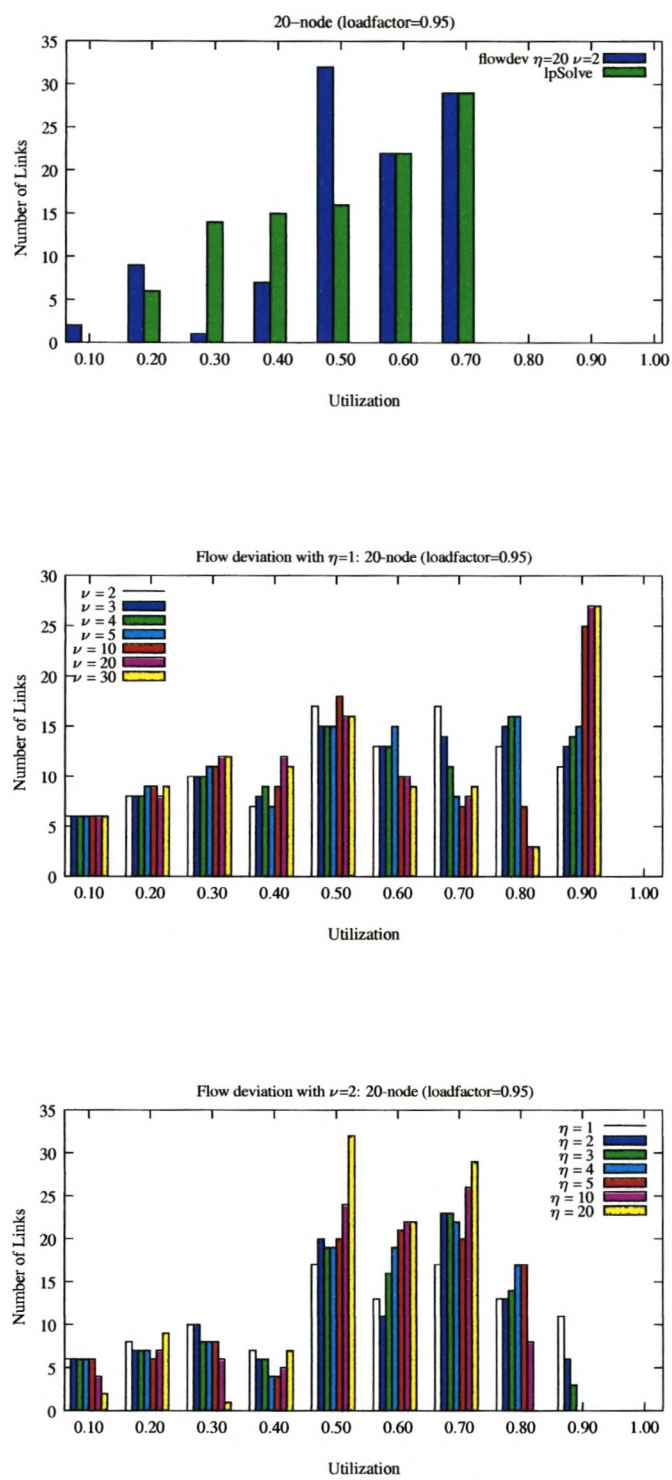


Figure 3.3: 20-node utilization bar charts

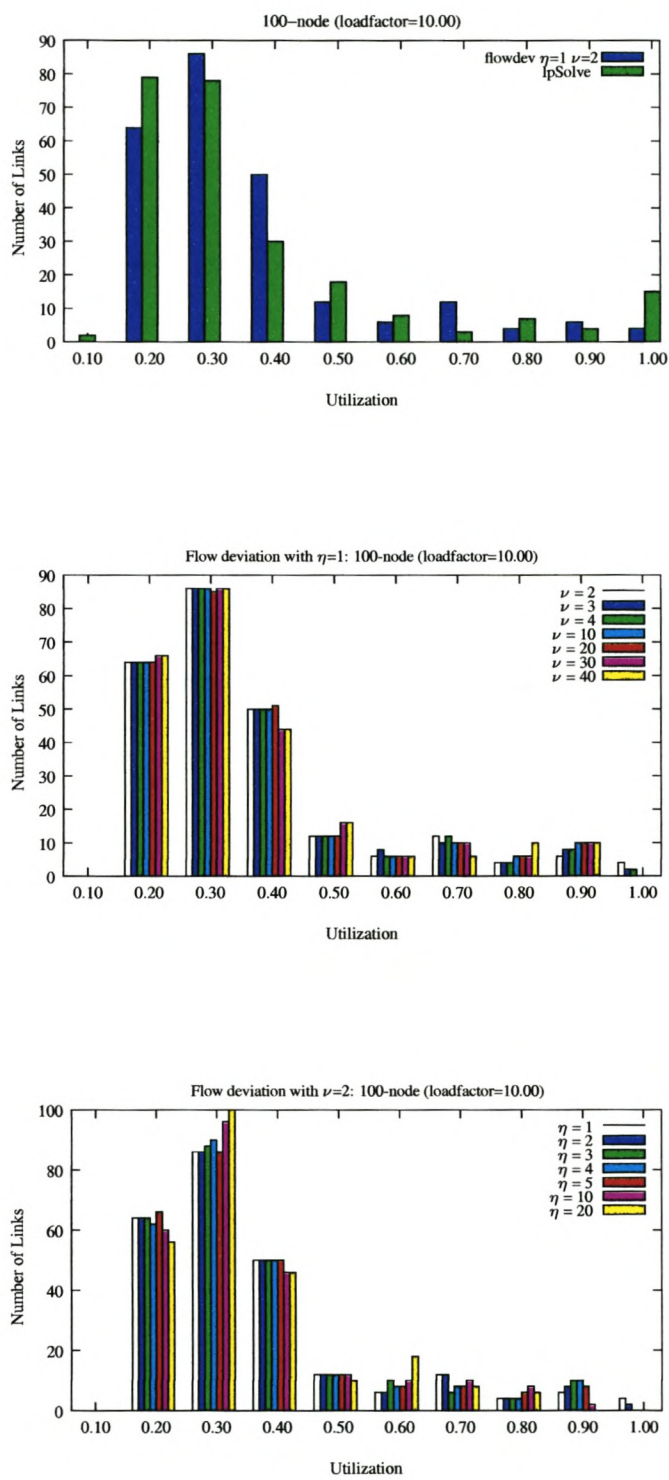


Figure 3.4: 100-node utilization bar charts



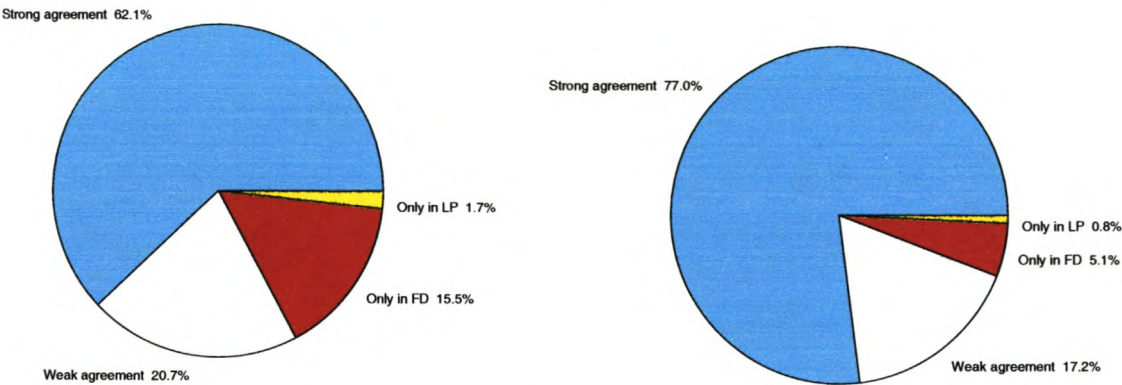


Figure 3.5: 10-node: (a) paths and (b) path flows

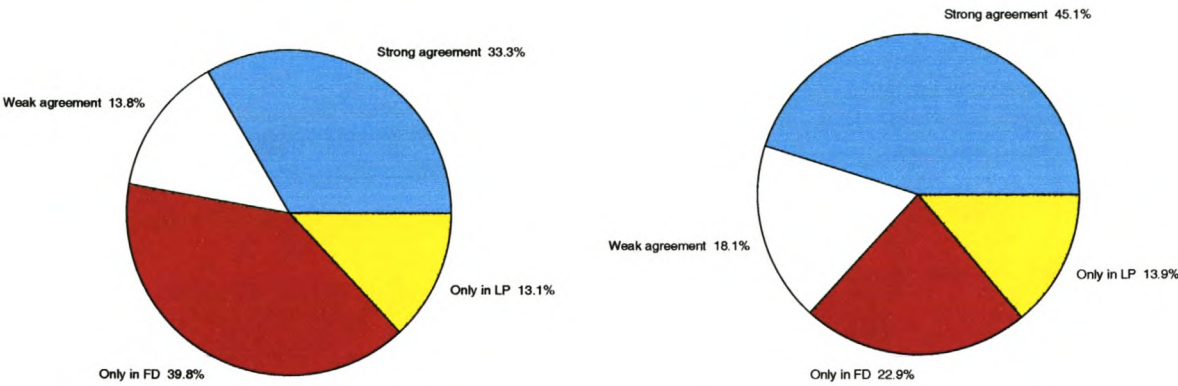


Figure 3.6: 20-node: (a) paths and (b) path flows

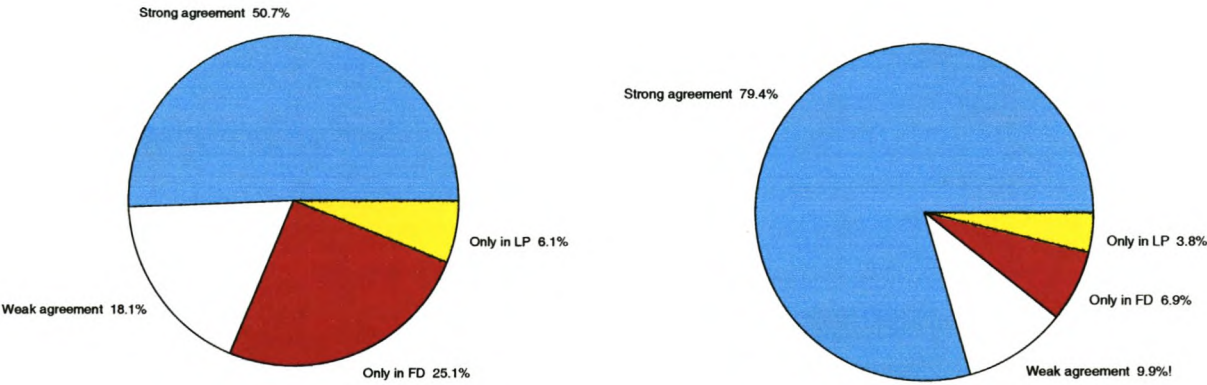


Figure 3.7: 100-node: (a) paths and (b) path flows

### 3.4 Conclusion

The more spare capacity available on links, the easier recovery mechanisms can restore the network throughput after a failure.

We formulated the path finding process in terms of constrained linear and nonlinear programming problems.

We investigated a linear objective function which can be used to find traffic engineered working paths such that the spare capacity on each link is maximized.

We presented and motivated our choice of a nonlinear objective function and an optimization method for finding traffic engineered paths.

We compared the distribution of the link utilizations of the linear and nonlinear methods and we saw that the nonlinear objective function can be parameterized so that it maximizes the spare capacity on each link.



## Chapter 4

# Computing a set of recovery paths: the Healing method

This chapter examines the effectiveness of using healed paths as recovery paths for a single uni-directional link failure. A healed path deploys a detour around the failed link without causing any loops in the path.

We use the flow deviation algorithm to find a set of optimal active paths and a set of backup paths for a network with no failures. We use these paths to compute an additional set of backup paths for a specific single uni-directional link failure and refer to these paths as healed paths. The flow deviation algorithm is then used on the failed network to determine which healed paths should be assigned bandwidth to obtain an optimal network configuration while the failed link is repaired.

The goal of this chapter is to evaluate the usefulness (or optimality) of different healed recovery pathsets for a single uni-directional link failure.

The remainder of this chapter is organized as follows. Section 4.1 introduces the flow deviation algorithms used in this thesis. The different recovery pathsets are explained in section 4.2. Section 4.3 explains the four experiments of this chapter. The results of the experiments are presented in section 4.4 and the conclusions are stated in section 4.5.

### 4.1 Flow Deviation Algorithms

The flow deviation algorithm finds optimal LSPs given a network with link capacities and traffic demands. We use the objective function (3.12) explained in the previous chapter with values  $\eta = 1$ ,  $\nu = 2$ ,  $\tau_i = 1$  and  $\sigma_i = \max(1.0, b_i/10.0)$ .

In each iteration of the algorithm a path-finding method is used to determine a set of shortest (least cost) paths to which a portion of flow from higher cost paths can be diverted. If the *pre-planned* (or local) path-finding method is used, only least cost paths that are already in the pre-planned pathset are considered. If the *reactive* (or global) path-finding method is used, new least cost paths may be considered and added to the existing pathset.

The standard flow deviation algorithm (FD), the Kleinrock algorithm, moves the same portion of flow from the higher cost paths to the least cost paths for all source-destination pairs. The amount of flow which is moved is computed by using a line search. The convergence rate of the FD algorithm is affected by congestion in the network. The more congested the network is, the more slowly the algorithm converges.

Another flow deviation version, the Bertsekas-Gallager flow deviation algorithm (BG), moves flow for one source-destination pair at a time. The amount of flow to move is computed directly rather than by performing a line search. The BG flow deviation algorithm is considered to be more efficient than the standard FD method because in most cases it converges faster [31].

## 4.2 Computing recovery pathsets

### The standard backup pathset

The standard backup pathset computed by the flow deviation method consists of all the paths that were identified by the flow deviation algorithm as candidates for inclusion in the active pathset but which were not included in the optimal solution. The backup paths are computed with no explicit failure scenarios in mind. Let  $\mathcal{B}$  denote the standard backup pathset.

### The KSP backup pathset

The KSP flow deviation algorithm is presented in [8, 34]. The  $K$ -shortest path (KSP) algorithm finds for each node pair  $(m, n)$  up to  $K_m$  link-disjoint shortest paths passing through the  $K_m$  nodes adjacent to the ingress node  $m$ . Two paths are said to be link-disjoint if they have no links in common.

The pre-planned BG flow deviation algorithm is used to partition the link-disjoint KSP pathset into an optimal active pathset and a backup pathset. Let  $\mathcal{B}_{\text{KSP}}$  denote the backup pathset.

Since the KSP paths are link-disjoint, the KSP method produces different pathsets depending upon the order used for the successive  $K$ -shortest path discovery: computing the link-disjoint paths passing



through the ingress neighbours selected in ascending, descending or random lexicographic order can produce different pathsets. This is illustrated by the network presented in figure 4.1. Although there are three paths from node 1 to node 4, the KSP method will find only one path if the path 1-2-3-4 is found first, otherwise the KSP method will find paths 1-5-3-4 and 1-2-6-4.

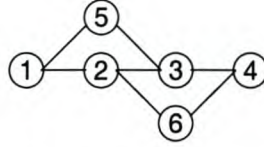


Figure 4.1: KSP path discrimination

### The healed backup pathset

The healed backup paths are computed with an explicit failure scenario in mind. In this chapter we consider a worst case uni-directional link failure as the failure scenario. A healed path deploys a short detour around the failed link without causing any loops in the path.

We use the KSP method to compute  $k$  link-disjoint detours around the failed link. The first detour is computed as the shortest possible detour in terms of the hop count. The healed backup pathset consists of all the failed active and standard backup paths that are healed using one of the  $k$  link-disjoint detour paths. One failed path can be used to find  $k$  healed backup paths, as long as no cycles occur in the path by replacing the failed link with one of the detour paths. Let  $\mathcal{H}$  denote the healed backup pathset. Let  $\mathcal{A}^{\mathcal{H}}$  denote the active pathset with the failed paths removed and replaced by healed paths, and let  $\mathcal{B}^{\mathcal{H}}$  denote the standard backup pathset with the failed paths removed and replaced by healed paths.

## 4.3 Explanation of the different experiments

Four experiments are presented in this section. Each experiment investigates the usefulness of a healed backup pathset in a network with a single uni-directional link failure. The healed backup pathset depends on the active and backup pathsets computed by the flow deviation method for a network with no failures.

In experiments 1, 2 and 4 the reactive BG flow deviation method is used to find an optimal active pathset  $\mathcal{A}$  and a backup pathset  $\mathcal{B}$  for a network with no failures. In experiment 3, we used the KSP algorithm to find a pre-planned link-disjoint pathset. The KSP flow deviation method could not find a feasible solution by using only the pre-planned pathset, but by adding a few paths to the pre-planned

pathset (by using the reactive method), a feasible solution  $\mathcal{A}_{\text{KSP}}$  and a backup pathset  $\mathcal{B}_{\text{KSP}}$  were found.

### Experiment 1

The reactive BG flow deviation algorithm with  $\mathcal{A}^{\mathcal{H}}$  as an initial input pathset was used to find an optimal active pathset for the network with a link failure.

### Experiment 2

The reactive BG flow deviation algorithm with  $\mathcal{A}^{\mathcal{H}} \cup \mathcal{B}^{\mathcal{H}}$  as an initial input pathset was used to find an optimal active pathset for the network with a link failure.

### Experiment 3

The reactive BG flow deviation algorithm with  $\mathcal{A}_{\text{KSP}}^{\mathcal{H}} \cup \mathcal{B}_{\text{KSP}}^{\mathcal{H}}$  as an initial input pathset was used to find an optimal active pathset for the network with a link failure.

### Experiment 4

The pre-planned BG flow deviation algorithm with  $\mathcal{A}^{\mathcal{H}} \cup \mathcal{B}^{\mathcal{H}}$  as an input pathset was used to find an optimal active pathset for the network with a link failure.

If a pathset is given as input to the flow deviation algorithm (as in the four experiments above), we normally would like to restrict the flow deviation method to find an optimal solution within the given pathset (i.e. we would use the pre-planned flow deviation method instead of the reactive method). However, experiments 1 to 3 have to use the reactive method because no feasible solution could be found using only the pre-planned pathset. Experiment 4 uses the pre-planned flow deviation method which means that no new paths can be added to the input pathset. Experiment 4 was able to find a feasible solution for the 50-node network but it could not find a feasible solution for the 100-node network.



## 4.4 Experimental results

The optimal BG flow deviation solution for the 50-node network with no failures reveals that the (20, 42)-link carries more flow than any other link in the network and that no other link in the network is used by more paths. We choose the (20, 42)-link as the worst case failure link and refer to it as link  $i$ . The active pathset  $\mathcal{A}$  consists of 3013 paths of which 119 paths use link  $i$ . The backup pathset  $\mathcal{B}$  consists of 818 paths of which 28 paths use link  $i$ . The KSP method finds three link-disjoint detours around the link  $i$  of which the shortest detour is 4 links long. The other detours (heals) are of length 5 and length 6 respectively.

The optimal BG flow deviation solution for the 100-node network with no failures reveals that the (54, 84)-link carries more flow than any other link in the network and that no other link in the network is used by more paths. We choose the (54, 84)-link as the worst case failure link and refer to it as link  $i$ . The active pathset  $\mathcal{A}$  consists of 10437 paths of which 1397 paths use link  $i$ . The backup pathset  $\mathcal{B}$  consists of 8674 paths of which 1963 paths use link  $i$ . The KSP method finds two link-disjoint detours around link  $i$ . Both detours are of length 15 which is the shortest possible heal for the (54, 84)-link. Note that the detours are excessively long so that the healed paths are not likely to function well as backup paths.

### 4.4.1 Comparison of the active paths before and after a link failure

The pie chart figures 4.2–4.8 compare the optimal active pathset  $\mathcal{A}$  for a network with no failures, versus the optimal active pathset  $\mathcal{A}_i$  for a network with link failure  $i$ .

Let  $r_1$  and  $r_2$  denote the same route (or LSP), where  $r_1 \in \mathcal{A}$  and  $r_2 \in \mathcal{A}_i$ . Let  $f_1$  and  $f_2$  denote the optimal bandwidth assignments for routes  $r_1$  and  $r_2$  respectively.

The pie charts on the left of figures 4.2–4.8 compare the routes. If

$$\frac{|f_1 - f_2|}{\max(f_1, f_2)} < 0.05$$

then the route is counted under the “strong agreement” pie chart slice, else the route is counted under the “weak agreement” slice.

The pie charts on the right of figures 4.2–4.8 compare the LSP bandwidth assignments. If

$$\frac{|f_1 - f_2|}{\max(f_1, f_2)} < 0.05$$

then the LSP bandwidth  $\max(f_1, f_2)$  is counted under the “strong agreement” pie chart slice, else  $\max(f_1, f_2)$  is counted under the “weak agreement” slice.

The pie charts reveal that most of the active paths before the link failure are also optimal active paths after the failure. These paths have before and after the link failure very similar optimal bandwidth assignments.

The pie chart slices “Only in set  $\mathcal{A}$ ” and “Only in set  $\mathcal{A}_i$ ” are of almost the same size for all the experiments. This means that the number of active paths not used after the link failure and the number of new active paths after the link failure does not differ much. The total amount of bandwidth assigned to the LSPs in each slice is also very much the same.

The pie charts do not show how many healed recovery paths were chosen as active paths after the failure. We analyze the “Only in set  $\mathcal{A}$ ” and the “Only in set  $\mathcal{A}_i$ ” pie chart slices in the next section.

#### 4.4.2 Performance of the healed recovery paths

Tables 4.1–4.7 summarise the experimental results.

In the top row of the table, the total number  $|\mathcal{A}|$  of active paths before and the total number  $|\mathcal{A}_i|$  of active paths after the link failure is given, as well as the number of computed healed recovery paths. In experiment 1 the standard backup pathset is not used which means that the healed pathset consists only of the healed active paths. In experiments 2 to 4 the total number of paths in the standard backup pathset is also given in the top row and the healed pathset consists of the healed active and backup paths.

In the bottom row of the table, the value of the objective function is given. This is the value computed by the flow deviation method for the network with a link failure. The lower the value, the better the flow deviation solution.

For the 50-node and the 100-node network the objective values of the first three experiments do not differ much from each other. The objective value of experiment 4 for the 50-node network is much higher than the other 3 experiments because the pre-planned flow deviation method was used and no new paths could be added to find a solution with a lower objective value.

Experiment 4 for the 50-node network is the only experiment which uses many of the healed recovery paths as active paths after the failure. All the other experiments prefer routes from the standard backup pathset and choose additional new routes to carry the traffic after the link failure.

Even though the healed recovery paths were computed with a specific failure scenario in mind and the standard backup paths were not, the optimization method does not prefer healed recovery paths over the paths in the standard backup pathset.



## 4.5 Conclusion

A method for computing a set of healed recovery paths for a specific single uni-directional link failure was presented and evaluated.

The healed recovery paths were not chosen by the flow deviation algorithm as optimal active paths after the link failure. New paths or paths from the standard backup pathset were preferred.

The healed paths could be used for quick recovery after a link failure, but for the two failure networks tested better results can be achieved by using new paths and paths from the standard backup pathset.

A possible reason why the healed recovery paths were not preferred is the fact that the detour paths around the failed link were many links long. The shortest possible heal for the failure link in the 100-node network was 15 links long and for the 50-node network the shortest heal was 4 links long. For both test networks an important link (which was used by the most paths and which carried a significant amount of flow) was chosen for failure. For an average link failure for which 2- or 3-link detours could be found, more of the healed recovery paths might be chosen as active paths by the flow deviation method, but in this chapter we only focussed on significant (or worst case) single link failures.

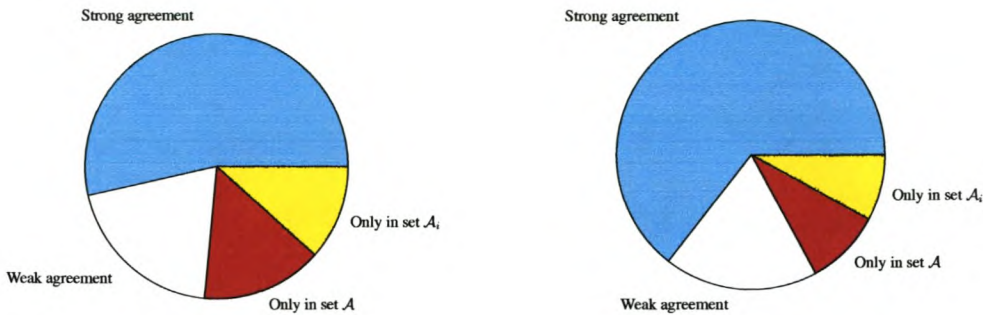


Figure 4.2: Experiment 1: (a) paths and (b) path bandwidths for the 50-node network

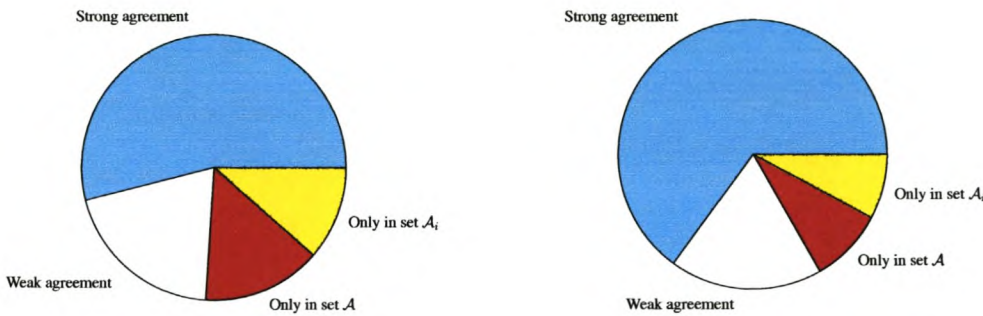


Figure 4.3: Experiment 2: (a) paths and (b) path bandwidths for the 50-node network

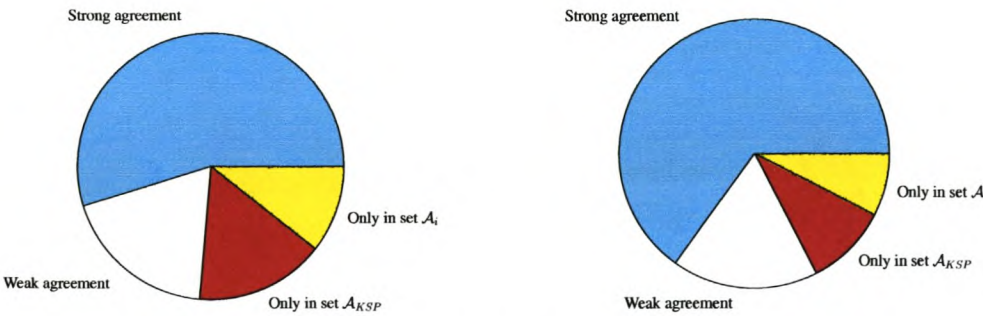


Figure 4.4: Experiment 3: (a) paths and (b) path bandwidths for the 50-node network

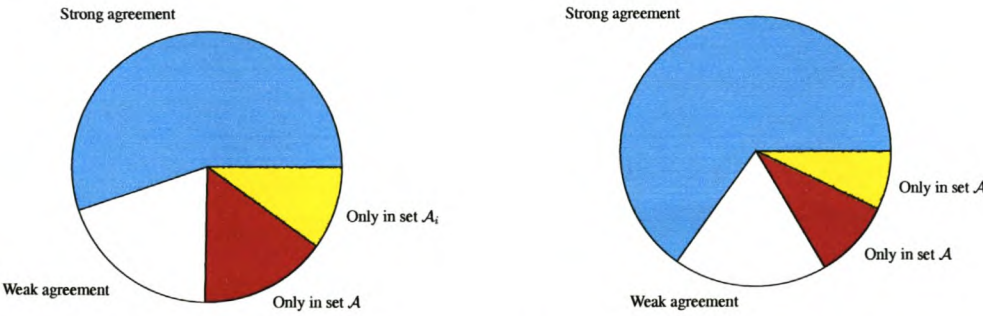


Figure 4.5: Experiment 4: (a) paths and (b) path bandwidths for the 50-node network



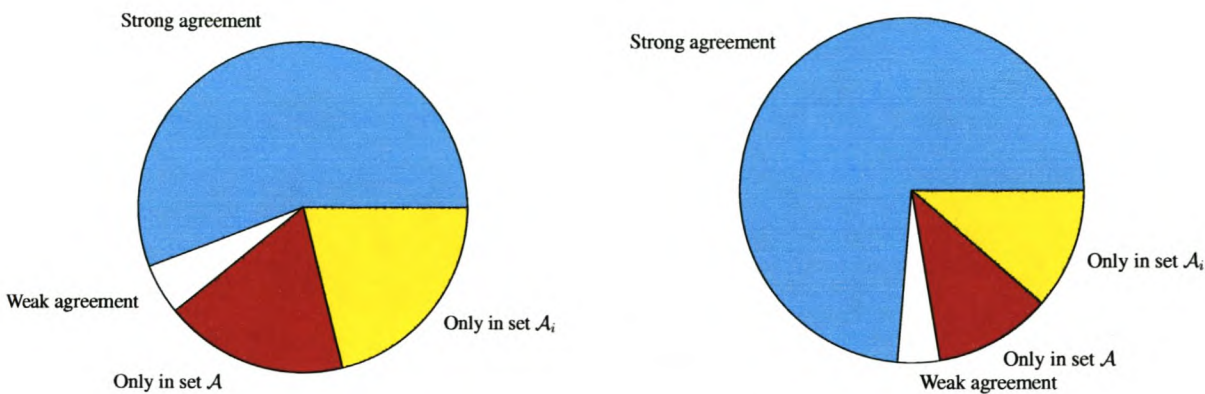


Figure 4.6: Experiment 1: (a) paths and (b) path bandwidths for the 100-node network

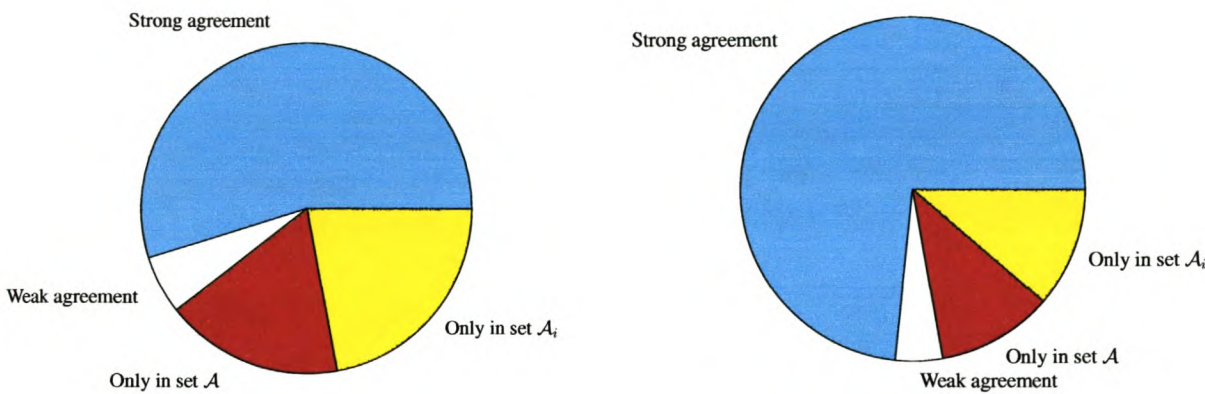


Figure 4.7: Experiment 2: (a) paths and (b) path bandwidths for the 100-node network

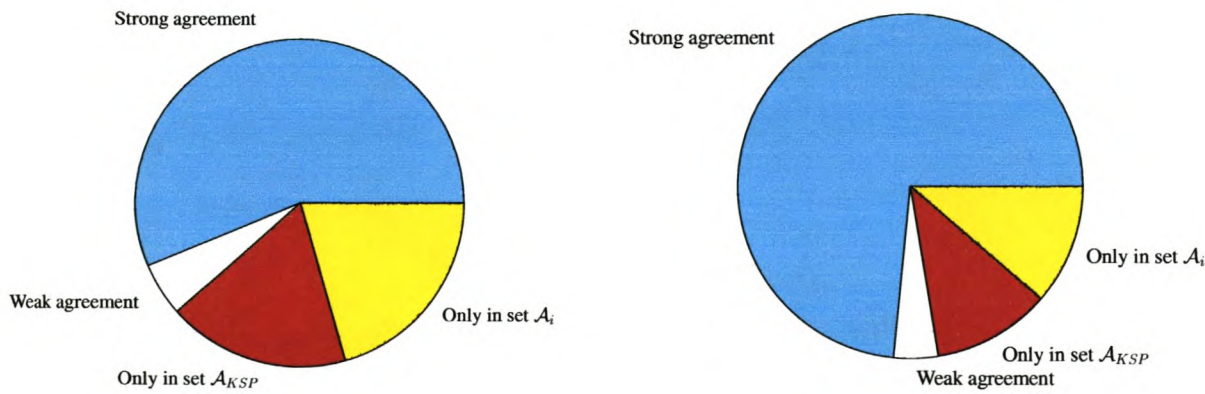


Figure 4.8: Experiment 3: (a) paths and (b) path bandwidths for the 100-node network

$ \mathcal{A}  = 3013$	$ \mathcal{A}_i  = 2901$	$ \mathcal{H}  = 231$
$ \mathcal{A} \cap \mathcal{A}_i $	= 2505	routes active before and after failure
	1824	routes in strong agreement
	681	routes in weak agreement
$ \mathcal{A} \setminus \mathcal{A}_i $	= 508	routes only in set $\mathcal{A}$
	119	routes in $\mathcal{A}$ failed
	389	routes in $\mathcal{A}$ not used after failure
$ \mathcal{A}_i \setminus \mathcal{A} $	= 396	routes only in set $\mathcal{A}_i$
$ \mathcal{A}_i \cap \mathcal{H} $	= 0	routes in $\mathcal{H}$ active after failure
	396	new routes active after failure
objective = 1.937745e+07		

Table 4.1: Results of experiment 1 for the 50-node network

$ \mathcal{A}  = 3013$	$ \mathcal{A}_i  = 2906$	$ \mathcal{H}  = 277$	$ \mathcal{B}  = 818$
$ \mathcal{A} \cap \mathcal{A}_i $	= 2517	routes active before and after failure	
	1835	routes in strong agreement	
	682	routes in weak agreement	
$ \mathcal{A} \setminus \mathcal{A}_i $	= 496	routes only in set $\mathcal{A}$	
	119	routes in $\mathcal{A}$ failed	
	377	routes in $\mathcal{A}$ not used after failure	
$ \mathcal{A}_i \setminus \mathcal{A} $	= 389	routes only in set $\mathcal{A}_i$	
$ \mathcal{A}_i \cap \mathcal{H} $	= 0	routes in $\mathcal{H}$ active after failure	
	205	new routes active after failure	
$ \mathcal{A}_i \cap \mathcal{B} $	= 184	routes in $\mathcal{B}$ active after failure	
objective = 1.937768e+07			

Table 4.2: Results of experiment 2 for the 50-node network



$ \mathcal{A}_{\text{KSP}}  = 3006$	$ \mathcal{A}_i  = 2834$	$ \mathcal{H}  = 737$	$ \mathcal{B}_{\text{KSP}}  = 6389$
$ \mathcal{A}_{\text{KSP}} \cap \mathcal{A}_i $	= 2477	routes active before and after failure	
	1843	routes in strong agreement	
	634	routes in weak agreement	
$ \mathcal{A}_{\text{KSP}} \setminus \mathcal{A}_i $	= 529	routes only in set $\mathcal{A}_{\text{KSP}}$	
	122	routes in $\mathcal{A}_{\text{KSP}}$ failed	
	407	routes in $\mathcal{A}_{\text{KSP}}$ not used after failure	
$ \mathcal{A}_i \setminus \mathcal{A}_{\text{KSP}} $	= 357	routes only in set $\mathcal{A}_i$	
$ \mathcal{A}_i \cap \mathcal{H} $	= 1	routes in $\mathcal{H}$ active after failure	
	116	new routes active after failure	
$ \mathcal{A}_i \cap \mathcal{B}_{\text{KSP}} $	= 240	routes in $\mathcal{B}_{\text{KSP}}$ active after failure	
objective = 1.937742e+07			

Table 4.3: Results of experiment 3 for the 50-node network

$ \mathcal{A}  = 3013$	$ \mathcal{A}_i  = 2834$	$ \mathcal{H}  = 297$	$ \mathcal{B}  = 818$
$ \mathcal{A} \cap \mathcal{A}_i $	= 2501	routes active before and after failure	
	1847	routes in strong agreement	
	654	routes in weak agreement	
$ \mathcal{A} \setminus \mathcal{A}_i $	= 512	routes only in set $\mathcal{A}$	
	119	routes in $\mathcal{A}$ failed	
	393	routes in $\mathcal{A}$ not used after failure	
$ \mathcal{A}_i \setminus \mathcal{A} $	= 333	routes only in set $\mathcal{A}_i$	
$ \mathcal{A}_i \cap \mathcal{H} $	= 121	routes in $\mathcal{H}$ active after failure	
	0	new routes active after failure	
$ \mathcal{A}_i \cap \mathcal{B} $	= 212	routes in $\mathcal{B}$ active after failure	
objective = 2.002956e+07			

Table 4.4: Results of experiment 4 for the 50-node network

$ \mathcal{A}  = 10437$	$ \mathcal{A}_i  = 10848$	$ \mathcal{H}  = 865$
$ \mathcal{A} \cap \mathcal{A}_i $	= 8048	routes active before and after failure
	7394	routes in strong agreement
	654	routes in weak agreement
$ \mathcal{A} \setminus \mathcal{A}_i $	= 2389	routes only in set $\mathcal{A}$
	1397	routes in $\mathcal{A}$ failed
	992	routes in $\mathcal{A}$ not used after failure
$ \mathcal{A}_i \setminus \mathcal{A} $	= 2800	routes only in set $\mathcal{A}_i$
$ \mathcal{A}_i \cap \mathcal{H} $	= 0	routes in $\mathcal{H}$ active after failure
	2800	new routes active after failure
objective = 1.833489e+06		

Table 4.5: Results of experiment 1 for the 100-node network

$ \mathcal{A}  = 10437$	$ \mathcal{A}_i  = 11063$	$ \mathcal{H}  = 1712$	$ \mathcal{B}  = 8674$
$ \mathcal{A} \cap \mathcal{A}_i $	= 8108	routes active before and after failure	
	7347	routes in strong agreement	
	761	routes in weak agreement	
$ \mathcal{A} \setminus \mathcal{A}_i $	= 2329	routes only in set $\mathcal{A}$	
	1397	routes in $\mathcal{A}$ failed	
	932	routes in $\mathcal{A}$ not used after failure	
$ \mathcal{A}_i \setminus \mathcal{A} $	= 2955	routes only in set $\mathcal{A}_i$	
$ \mathcal{A}_i \cap \mathcal{H} $	= 0	routes in $\mathcal{H}$ active after failure	
	1744	new routes active after failure	
$ \mathcal{A}_i \cap \mathcal{B} $	= 1211	routes in $\mathcal{B}$ active after failure	
objective = 1.833505e+06			

Table 4.6: Results of experiment 2 for the 100-node network



$ \mathcal{A}_{\text{KSP}}  = 10438$		$ \mathcal{A}_i  = 10788$		$ \mathcal{H}  = 2733$		$ \mathcal{B}_{\text{KSP}}  = 18349$	
$ \mathcal{A}_{\text{KSP}} \cap \mathcal{A}_i $		=	8086	routes active before and after failure			
			7394	routes in strong agreement			
			692	routes in weak agreement			
$ \mathcal{A}_{\text{KSP}} \setminus \mathcal{A}_i $		=	2352	routes only in set $\mathcal{A}_{\text{KSP}}$			
			1393	routes in $\mathcal{A}_{\text{KSP}}$ failed			
			959	routes in $\mathcal{A}_{\text{KSP}}$ not used after failure			
$ \mathcal{A}_i \setminus \mathcal{A}_{\text{KSP}} $		=	2702	routes only in set $\mathcal{A}_i$			
$ \mathcal{A}_i \cap \mathcal{H} $		=	3	routes in $\mathcal{H}$ active after failure			
			1403	new routes active after failure			
$ \mathcal{A}_i \cap \mathcal{B}_{\text{KSP}} $		=	1296	routes in $\mathcal{B}_{\text{KSP}}$ active after failure			
objective = 1.833539e+06							

Table 4.7: Results of experiment 3 for the 100-node network

## Chapter 5

# Computing a set of recovery paths: the $\mathcal{B}^{\mathcal{F}}$ method

This chapter presents and evaluates a method for finding a set of traffic engineered recovery paths by using the flow deviation algorithm.

The proposed method can be applied to any set of failure scenarios. For illustrative purposes, single link failure scenarios are chosen as the most probable failure events, because according to the literature (see [41] and the references therein) a complete fibre cut is the most common and frequently reported failure event among disastrous network failures over the decades.

The remainder of this chapter is organized as follows. Section 5.1 presents the proposed method of computing a set of traffic engineered recovery paths. Section 5.2 states the characteristics of our protection model. Section 5.3 presents the experimental results and the conclusions are stated in section 5.4.

### 5.1 Computing the set of $\mathcal{B}^{\mathcal{F}}$ recovery paths

The following method, presented in [35], was used to compute a set of traffic engineered recovery paths.

Let  $\mathcal{F}$  denote a set of network failure scenarios. The flow deviation method was first used to find a set  $\mathcal{A}$  of optimal LSPs for the network in the absence of failures. For each failure scenario  $i \in \mathcal{F}$  the flow deviation method was then used to find a set  $\mathcal{A}_i$  of optimal LSPs for the network with failure  $i$ . The set  $\mathcal{B}_i = \mathcal{A}_i \setminus \mathcal{A}$  defines the set of optimal recovery paths for failure  $i$ , and the set  $\mathcal{B}^{\mathcal{F}} = \cup_{i \in \mathcal{F}} \mathcal{B}_i$  defines the optimal set of recovery paths for all the failure scenarios.



## 5.2 Characteristics of our protection model

Our recovery (or protection) model makes use of *split path protection* where multiple recovery paths can carry the traffic of a working counterpart. This is useful and sometimes necessary when no single recovery path can be found that can carry the entire traffic of the working path. Furthermore, to make optimal use of resources and balance the load on the links (or maximize the link slacks), it is sometimes necessary to split the traffic of a failed working path over more than one recovery path although this may be detrimental to the performance of the restored traffic.

Our protection model is subject to the traffic engineering goal of optimal use of resources (see goal (1) of section 2.1.2) in the sense that when the link loads are low, the flow deviation solution yields the OSPF routes, but when the link loads are high, the flow deviation solution selects relatively short routes so that the slack on each link is maximized. This means that goal (3) is also satisfied, since the network reliability and availability are maximized by not overloading any one specific link. Our protection recovery model may be applicable for an entire end-to-end path or for segments of an end-to-end path (goal 6). The protection model does not trigger lower layer protection switching (goal 7) and aims to minimize the loss of data (goal 8). Our protection recovery model works with pre-established reserved-on-demand recovery paths. Some of these paths are limited recovery paths only when the network is not well designed and capacitated, otherwise all of the recovery paths are equivalent recovery paths. If the network is sufficiently capacitated then the recovery paths can meet the resource requirements of the working paths and achieve the same performance characteristics as the working paths (goal 10).

## 5.3 Experimental results

This section presents results for the 20-, 50- and the two 100-node networks described in appendix A.

For the two 100-node networks a load factor of three was used. This means that a total of 75,000 units of flow were offered to the 100-node networks and not a total of 25,000 units as described in appendix A.

The parameters of the flow deviation penalty function (3.14) are  $\eta = 1$ ,  $\nu = 1$ ,  $\tau_i = 1$  and  $\sigma_i = b_i/10$ .

A single bi-directional link failure between source-destination pair  $(s, d)$  means that both the uni-directional link  $(s, d)$  and the uni-directional link  $(d, s)$  fail. A single uni-directional link failure means that only the uni-directional link  $(s, d)$  fails in the network.



	$ \mathcal{A} $	Worst case			Average	
		$ \mathcal{A}_i $	$ \mathcal{B}_i $	$ \mathcal{B}^{\mathcal{F}} $	$ \mathcal{A}_i $	$ \mathcal{B}_i $
20-node (uni)	540	613	144	1189	537	35
20-node (bi)	540	717	275	1204	546	60
50-node (uni)	3903	3335	211	5427	3765	111
50-node (bi)	3903	3379	373	5272	3690	161
100-node (uni)	13984	11062	1447	45648	12969	688
100-node (bi)	13984	10384	2910	122278	12941	1054
planar 100-node (uni)	15090	15266	4237	571530	15218	1216
planar 100-node (bi)	15090	15332	6748	346868	15276	1476

Table 5.1: Number of paths in  $\mathcal{A}$ ,  $\mathcal{A}_i$ ,  $\mathcal{B}_i$  and  $\mathcal{B}^{\mathcal{F}}$  for single link failure scenarios (uni- and bi-directional).

A worst case failure scenario for each test network is chosen as one in which a link fails which transports more flow than any other link and which is used by the most active LSPs. This link is usually one of the most heavily utilized links.

Table 5.1 presents a summary of the number  $|\mathcal{A}|$  of paths found by the flow deviation method when no failures are present and the number  $|\mathcal{A}_i|$  of paths when a worst case failure event  $i$  occurs. The average number of paths in  $\mathcal{A}_i$  and  $\mathcal{B}_i$  after all possible single link failures are also given, as well as the number of paths in  $\mathcal{B}^{\mathcal{F}}$  that would be needed to cover all possible single link failure scenarios. Thus for a network with  $L$  links, the average  $|\mathcal{A}_i| = \sum_{i=1}^L |\mathcal{A}_i| / L$ .

The number of paths in  $\mathcal{B}^{\mathcal{F}}$  is very large – especially for the planar 100-node network uni-directional link failure scenarios where a total of 571,530 back-up paths are found. Although the flow deviation method finds optimal active path sets for all link failure scenarios, these sets are not unique (see appendix B). It is possible, for example, that for two link failures  $i$  and  $j$  the optimal sets  $\mathcal{A}_i$  and  $\mathcal{A}_j$  found by flow deviation could be chosen differently to result in two sets that have more paths in common. This will result in less paths included in  $\mathcal{B}^{\mathcal{F}}$ .

Table 5.2 gives a summary of the percentage of active paths broken and the percentage of the total offered traffic affected by a single link failure. The percentages in the average case compared to the percentages in the worst case scenario is much lower – so much lower that we can conclude that many of the single link failures do not have a significant effect on the network. In the average case for the planar 100-node network only 1.2% of all the active paths fail and only 0.8% of the offered traffic is affected in a single uni-directional link failure scenario. It would be more appropriate to compute a set  $\mathcal{B}^{\mathcal{F}}$  to cover only significant link failure scenarios and not all possible link failure scenarios because the  $\mathcal{B}^{\mathcal{F}}$  set of paths would then be smaller and more manageable.



	Worst case: percentage of paths broken    traffic affected		Average: percentage of paths broken    traffic affected	
20-node (uni)	7.4	8.1	2.4	2.0
20-node (bi)	14.8	14.0	4.7	4.0
50-node (uni)	4.0	4.1	1.6	1.4
50-node (bi)	8.0	7.8	3.2	2.9
100-node (uni)	19.9	6.7	4.0	2.6
100-node (bi)	39.9	13.4	8.1	5.2
planar 100-node (uni)	17.0	7.4	1.2	0.8
planar 100-node (bi)	33.9	14.7	2.4	1.5

Table 5.2: Percentage of active paths broken and percentage of total offered traffic affected by a single link failure scenario (uni- and bi-directional).

The performance of the recovery paths is presented in tables 5.3 – 5.10 and also in graphical form as pie charts. Each pie chart has five slices where

- “ $\mathcal{A}$ -broken” denotes the set of paths in  $\mathcal{A} \setminus \mathcal{A}_i$  which fail when link  $i$  failed, where  $\mathcal{A}$  denotes the optimal set of active paths prior to the link failure and  $\mathcal{A}_i$  denotes the optimal set of active paths after a single link  $i$  failed,
- “ $\mathcal{A}$ -not\_used” denotes the set of paths in  $\mathcal{A} \setminus \mathcal{A}_i$  which did not fail but were no longer used after link  $i$  failed,
- “Strong agreement” denotes the set of paths in  $\mathcal{A} \cap \mathcal{A}_i$  which survived the link failure and whose bandwidth assignments before and after link  $i$  failed differ by less than 5%,
- “Weak agreement” denotes the set of paths  $\mathcal{A} \cap \mathcal{A}_i$  which survived the link failure and whose bandwidth assignments before and after link  $i$  failed differ by more than 5%,
- “ $B^F$ -active” denotes the set of paths  $B^F \cap \mathcal{A}_i$  that were deployed when link  $i$  failed.

With respect to figures 5.1 – 5.16 note that the figures (a) on the left display the number of paths in each slice; the figures (b) on the right display the sum of the bandwidths  $B_P$  of all the paths  $P$  in each slice-set. In the “strong” and “weak agreement” slices, the sum of  $\max(B_P \text{ before failure}, B_P \text{ after failure})$  is displayed.

The bottom two pie charts on pages 54–61 present the performance of the recovery paths averaged over all single link failures. The standard deviation of the measure displayed by each slice is represented by the area of the segment projecting beyond the pie chart circumference.

	Worst case		Average (StdDev)			
	paths	bandwidths	paths		bandwidths	
$\mathcal{A}$ -broken	40	107974	12.7	(9.5)	26593	(18615)
$\mathcal{A}$ -not_used	31	14578	25.7	(10.0)	4956	(7141)
strong agreement	285	952665	358.9	(41.5)	1141085	(88048)
weak agreement	184	323972	142.5	(30.6)	195580	(86426)
$B^F$ -active	144	126710	35.4	(29.8)	28337	(25616)

Table 5.3: Commonality among the paths and path bandwidths used before and after uni-directional link failures in the 20-node network

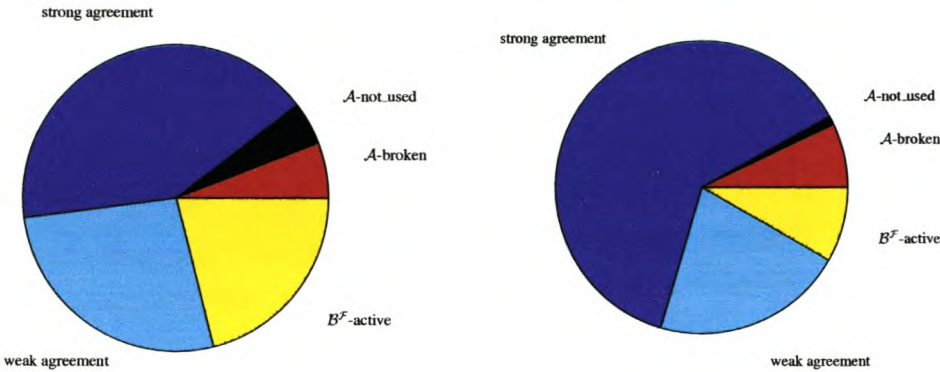


Figure 5.1: Commonality among the (a) paths and (b) path bandwidths used before and after a worst case uni-directional link failure in the 20-node network

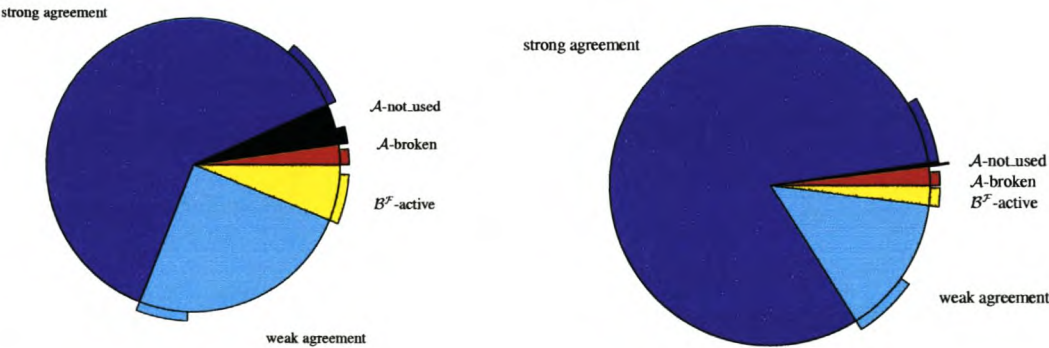


Figure 5.2: Average and standard deviation of the commonality among the (a) paths and (b) path bandwidths used before and after 102 uni-directional link failure scenarios in the 20-node network



	Worst case		Average (StdDev)			
	paths	bandwidths	paths		bandwidths	
$\mathcal{A}$ -broken	80	186801	25.5	(18.4)	53187	(36576)
$\mathcal{A}$ -not_used	18	17959	28.7	(10.9)	8140	(10838)
strong agreement	257	855797	327.5	(44.8)	1066086	(104444)
weak agreement	185	364839	158.1	(29.7)	258179	(92159)
$B^F$ -active	275	215867	60.4	(50.8)	55493	(46005)

Table 5.4: Commonality among the paths and path bandwidths used before and after bi-directional link failures in the 20-node network

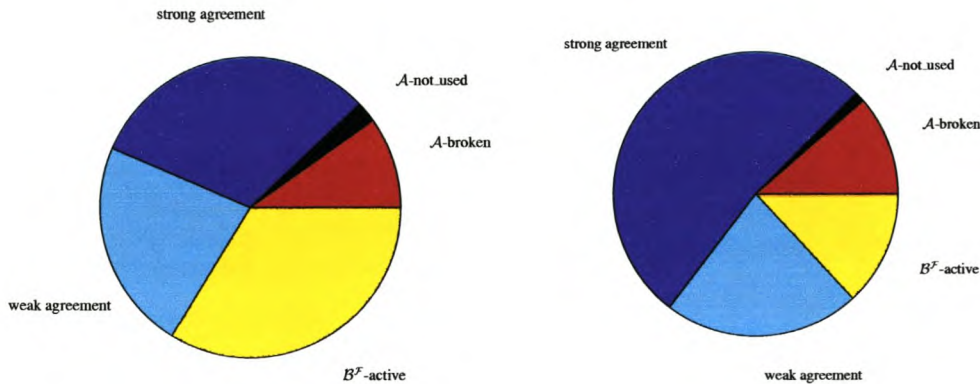


Figure 5.3: Commonality among the (a) paths and (b) path bandwidths used before and after a worst case bi-directional link failure in the 20-node network

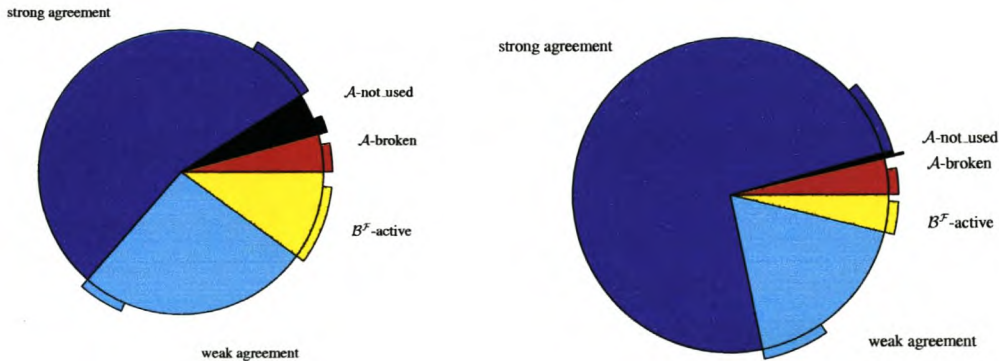


Figure 5.4: Average and standard deviation of the commonality among the (a) paths and (b) path bandwidths used before and after 51 bi-directional link failure scenarios in the 20-node network

	Worst case		Average (StdDev)			
	paths	bandwidths	paths		bandwidths	
$\mathcal{A}$ -broken	156	266548	62.4	(20.2)	93954	(36243)
$\mathcal{A}$ -not_used	623	205161	185.8	(133.1)	22208	(21339)
strong agreement	1823	4778890	2175.5	(172.5)	5315354	(155940)
weak agreement	1301	1895113	1479.1	(140.6)	1404397	(176827)
$\mathcal{B}^{\mathcal{F}}$ -active	211	266039	110.6	(31.7)	68162	(33634)

Table 5.5: Commonality among the paths and path bandwidths used before and after uni-directional link failures in the 50-node network

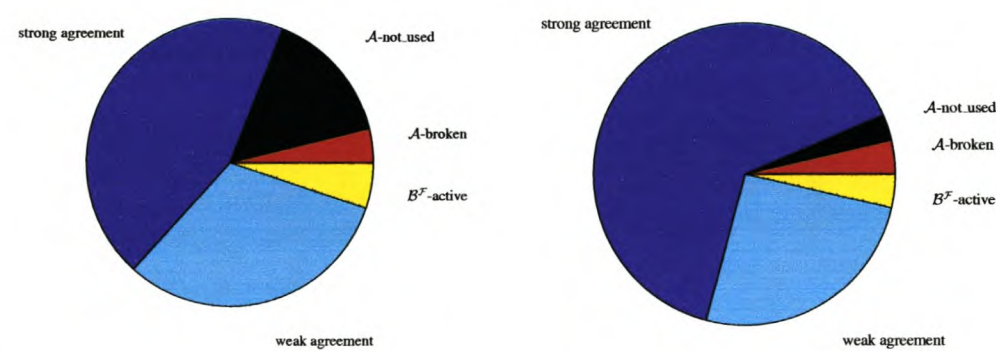


Figure 5.5: Commonality among the (a) paths and (b) path bandwidths used before and after a worst case uni-directional link failure in the 50-node network

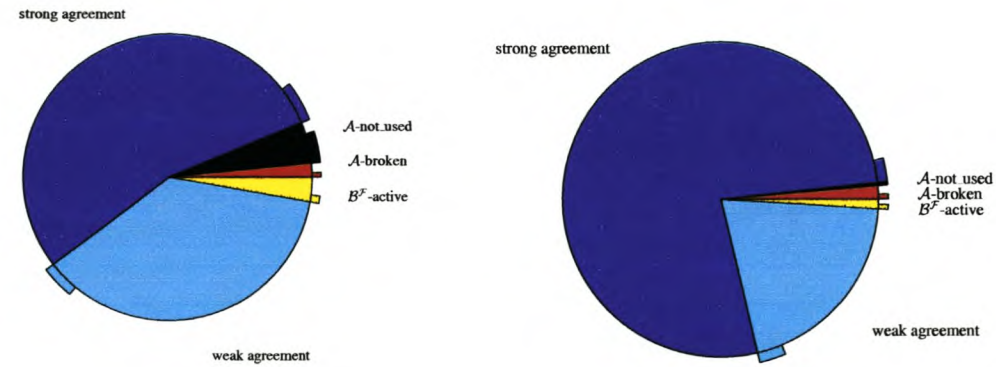


Figure 5.6: Average and standard deviation of the commonality among the (a) paths and (b) path bandwidths used before and after 202 uni-directional link failure scenarios in the 50-node network



	Worst case		Average (StdDev)			
	paths	bandwidths	paths		bandwidths	
$\mathcal{A}$ -broken	312	513710	124.8	(39.8)	187909	(70299)
$\mathcal{A}$ -not_used	585	195138	249.5	(139.6)	36109	(29308)
strong agreement	1748	4589110	2053.3	(145.1)	5128410	(175688)
weak agreement	1258	1889497	1475.2	(127.7)	1570555	(172697)
$B^{\mathcal{F}}$ -active	373	445708	161.3	(51.1)	131192	(63025)

Table 5.6: Commonality among the paths and path bandwidths used before and after bi-directional link failures in the 50-node network

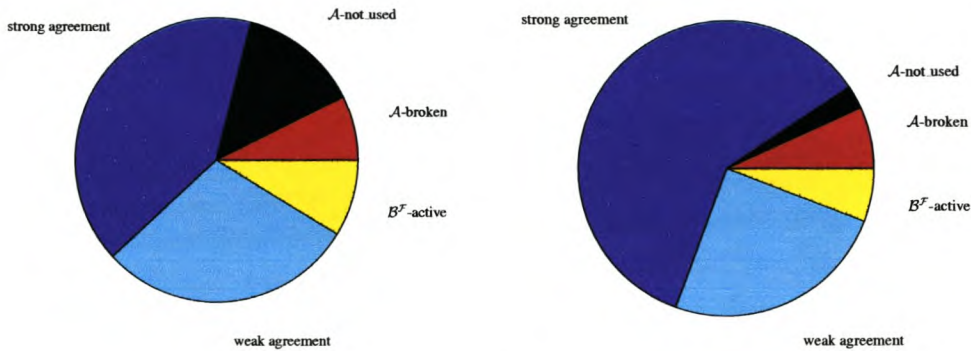


Figure 5.7: Commonality among the (a) paths and (b) path bandwidths used before and after a worst case bi-directional link failure in the 50-node network

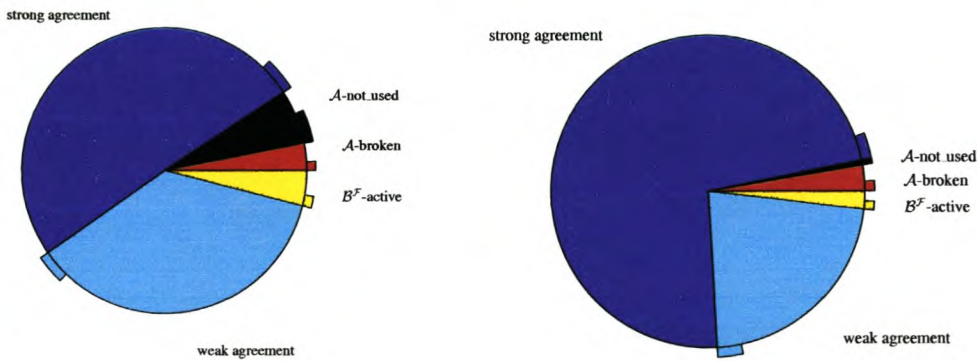


Figure 5.8: Average and standard deviation of the commonality among the (a) paths and (b) path bandwidths used before and after 101 bi-directional link failure scenarios in the 50-node network

	Worst case		Average (StdDev)			
	paths	bandwidths	paths		bandwidths	
$\mathcal{A}$ -broken	2788	5023	564.3	(485.4)	1931	(1318)
$\mathcal{A}$ -not_used	1581	597	1138.0	(1039.3)	415	(435)
strong agreement	7929	65425	9261.4	(1782.0)	67648	(3874)
weak agreement	1686	5015	3020.1	(1634.3)	6213	(3828)
$B^{\mathcal{F}}$ -active	1447	4947	687.6	(646.9)	1838	(1363)

Table 5.7: Commonality among the paths and path bandwidths used before and after uni-directional link failures in the 100-node network

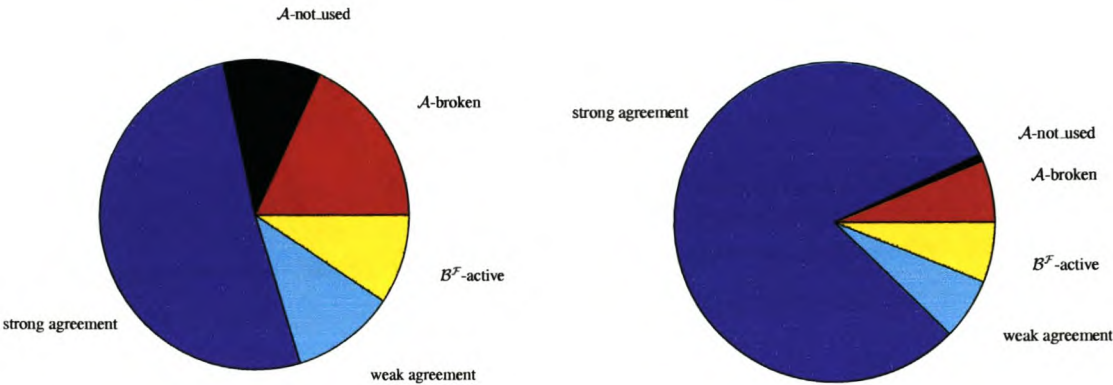


Figure 5.9: Commonality among the (a) paths and (b) path bandwidths used before and after a worst case uni-directional link failure in the 100-node network

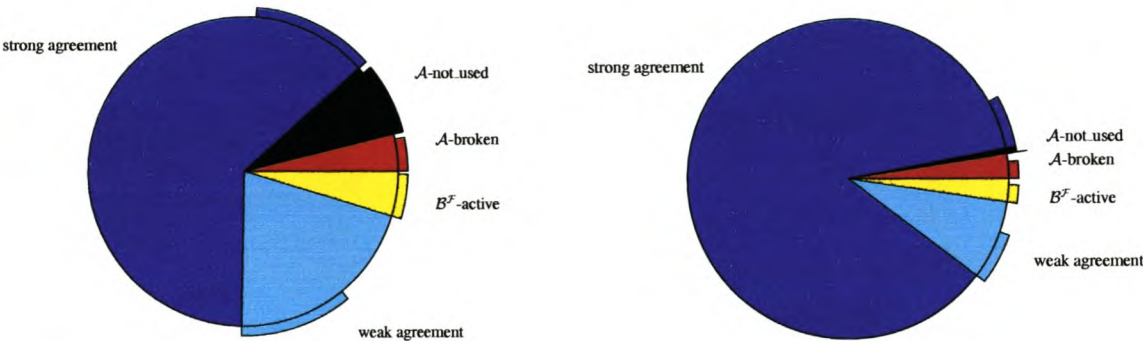


Figure 5.10: Average and standard deviation of the commonality among the (a) paths and (b) path bandwidths used before and after 236 uni-directional link failure scenarios in the 100-node network



	Worst case		Average (StdDev)			
	paths	bandwidths	paths		bandwidths	
$\mathcal{A}$ -broken	5576	10047	1127.9	(977.7)	3866	(2664)
$\mathcal{A}$ -not_used	934	848	968.9	(989.1)	380	(476)
strong agreement	6530	61226	9135.7	(2222.9)	66198	(4706)
weak agreement	944	4420	2751.2	(1553.9)	5748	(3824)
$\mathcal{B}^{\mathcal{F}}$ -active	2910	10021	1054.1	(978.0)	3617	(2698)

Table 5.8: Commonality among the paths and path bandwidths used before and after bi-directional link failures in the 100-node network

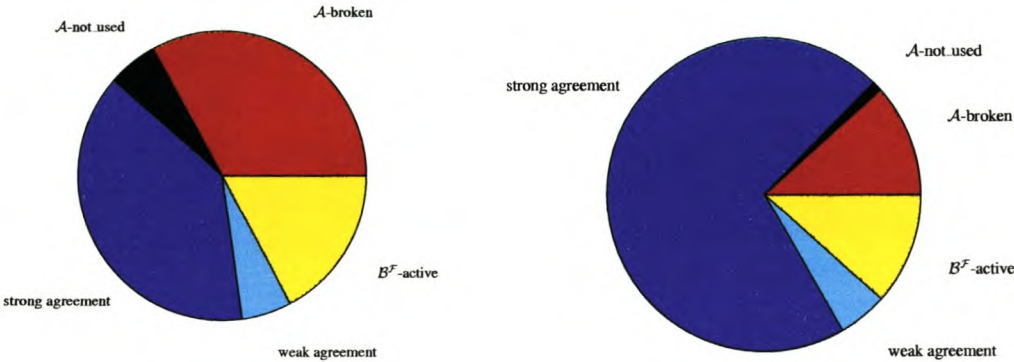


Figure 5.11: Commonality among the (a) paths and (b) path bandwidths used before and after a worst case bi-directional link failure in the 100-node network

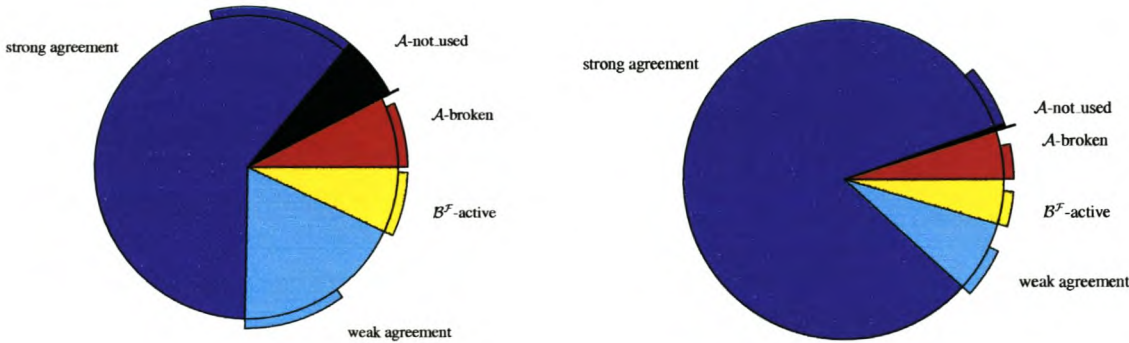


Figure 5.12: Average and standard deviation of the commonality among the (a) paths and (b) path bandwidths used before and after 116 bi-directional link failure scenarios in the 100-node network

	Worst case		Average (StdDev)			
	paths	bandwidths	paths	bandwidths	paths	bandwidths
$\mathcal{A}$ -broken	2559	5512	178.1	(242.7)	571	(699)
$\mathcal{A}$ -not_used	1502	844	909.6	(214.1)	211	(125)
strong agreement	6320	55296	9064.2	(571.6)	66498	(1776)
weak agreement	4709	15921	4937.9	(415.1)	8774	(1464)
$B^F$ -active	4237	6885	1216.0	(595.5)	673	(793)

Table 5.9: Commonality among the paths and path bandwidths used before and after uni-directional link failures in the planar 100-node network

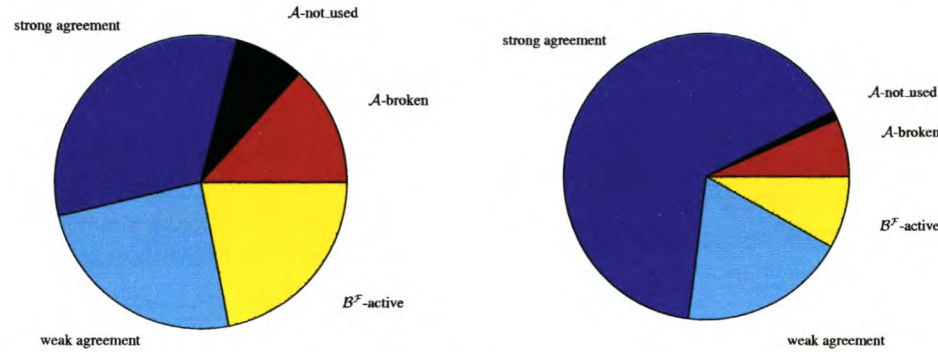


Figure 5.13: Commonality among the (a) paths and (b) path bandwidths used before and after a worst case uni-directional link failure in the planar 100-node network

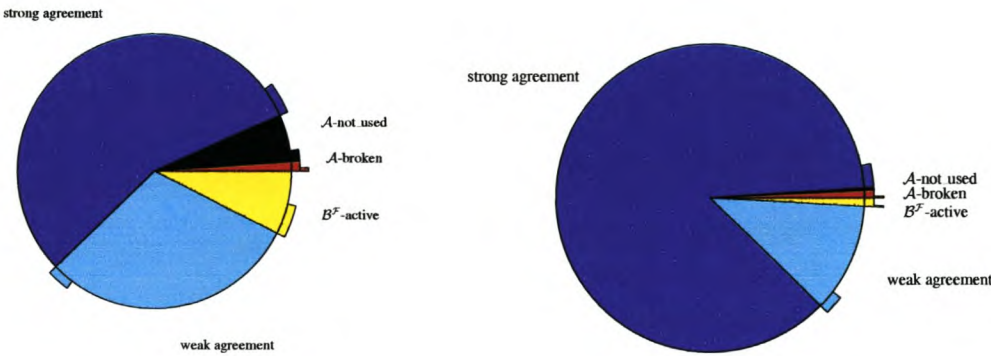


Figure 5.14: Average and standard deviation of the commonality among the (a) paths and (b) path bandwidths used before and after 470 uni-directional link failure scenarios in the planar 100-node network



	Worst case		Average (StdDev)			
	paths	bandwidths	paths		bandwidths	
$\mathcal{A}$ -broken	5118	11025	356.3	(486.0)	1142	(1400)
$\mathcal{A}$ -not_used	1388	953	934.1	(250.0)	244	(157)
strong agreement	5158	51227	8843.7	(620.2)	65682	(2460)
weak agreement	3426	14930	4955.7	(413.1)	9150	(1652)
$\mathcal{B}^{\mathcal{F}}$ -active	6748	12115	1476.0	(933.4)	1194	(1477)

Table 5.10: Commonality among the paths and path bandwidths used before and after bi-directional link failures in the planar 100-node network

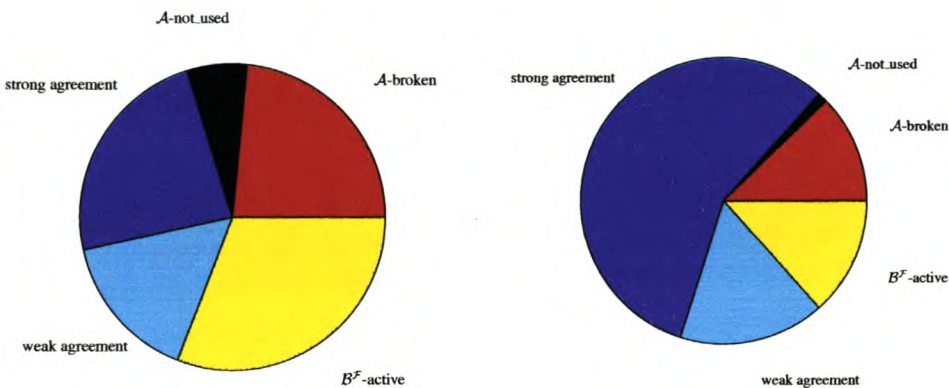


Figure 5.15: Commonality among the (a) paths and (b) path bandwidths used before and after a worst case bi-directional link failure in the planar 100-node network

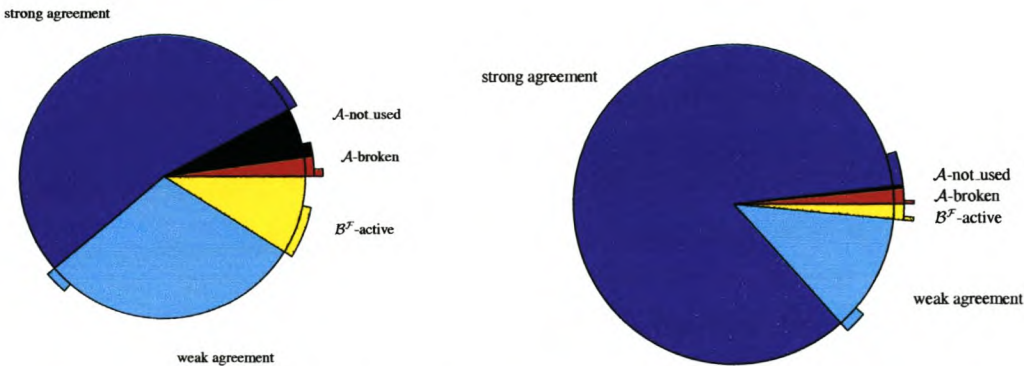


Figure 5.16: Average and standard deviation of the commonality among the (a) paths and (b) path bandwidths used before and after 235 bi-directional link failure scenarios in the planar 100-node network

	Worst case failure	objective
20-node	none	2.316670e+06
20-node (uni)	(4,15)	2.472958e+06
20-node (bi)	(4,15)(15,4)	2.581132e+06
50-node	none	1.956481e+07
50-node (uni)	(20,42)	1.984852e+07
50-node (bi)	(20,42)(42,20)	2.011471e+07
100-node	none	5.501851e+05
100-node (uni)	(36,62)	5.771163e+05
100-node (bi)	(36,62)(62,36)	6.040424e+05
planar 100-node	none	2.878131e+05
planar 100-node (uni)	(17,50)	3.032226e+05
planar 100-node (bi)	(17,50)(50,17)	3.185968e+05

Table 5.11: The value of the objective function before and after the worst case single link failure scenarios (uni- and bi-directional).

Table 5.11 presents a summary of the objective function values found by the flow deviation method when no failures are present and when a worst case failure event occurs. When no failures are present the objective value is at its lowest. The network objective function increases when a uni-directional link failure occurs, and it increases even more when a bi-directional link failure is experienced.

The specific worst case single uni- and bi-directional links which were chosen for failure for each network is also given in the above table.

## 5.4 Conclusion

The method of computing a recovery pathset presented in this chapter not only yields a sufficient number of recovery paths to cover specific failure scenarios, but the recovery paths also satisfy the traffic engineering objectives which are encoded into the flow deviation objective function. Thus should the network failure persist for a significant time, using these traffic engineered back-up paths will result in an optimal traffic distribution and a high network reliability because the objective function chose paths so as not to overload any one specific link.

The drawback of this method is that the number of paths in  $\mathcal{B}^{\mathcal{F}}$  can be very large because it depends on the number and nature of the different failure scenarios chosen to be protected. The more failure scenarios, the more paths are in  $\mathcal{B}^{\mathcal{F}}$ . The different failure scenarios should therefore be chosen sparingly.



## Chapter 6

# Conclusion

This thesis considered the problem of finding an in some sense optimal set of pre-established traffic engineered recovery paths, given a network with link capacities and traffic demands.

We showed that our choice of a nonlinear objective function and optimization method can find traffic engineered working paths such that the spare capacity on each link is maximized. The more spare capacity available in a network, the easier the recovery mechanisms can restore the network throughput after a failure.

We presented and evaluated two simple methods of computing a set of recovery paths. The healing method was used to find a set of healed recovery paths for a specific worst case single link failure scenario. For the networks tested, the healed recovery paths were not chosen as optimal active paths, possibly because the shortest heal (the detour around the failed link) used many links for the worst case link failure scenario.

The second protection method computed a recovery pathset for a set of possible failure scenarios. These recovery paths satisfied the traffic engineering objectives which were encoded into the objective function. Thus should the network failure persist for a significant time, using these recovery paths would result in an optimal traffic distribution and a high network reliability because the objective function had chosen paths so as not to overload any one specific link. However, the number of paths in the recovery pathset can be very large if the set of important failure scenarios to be protected against was not chosen sparingly.

## Appendix A

# Network models

This appendix describes the 10-, 20-, 50- and two 100-node network models used in this thesis.

Obtaining realistic network data for testing purposes can be a troublesome task because network service providers are unwilling to reveal the details of their networks due to competition in the telecommunication industry.

This thesis used network data generated by Villamizar [50]. These network models do not have fixed node placements. The *VCG visualization tool* [39] was used to produce figures A.1 to A.4 with the minimum number of edge crossings.

All the edges (or links) shown on the figures represent two uni-directional links, one in each direction.

A description of the models with their link capacities and offered traffics can be found at the URL <http://www.cs.sun.ac.za/projects/COE/models.zip>.

### The 10-node network

The 10-node network has 58 uni-directional links and carries 1 traffic class. The links are capacitated with 1,063,065 units of bandwidth. A total of 386,211 units of flow are offered to the 90 source-destination pairs.



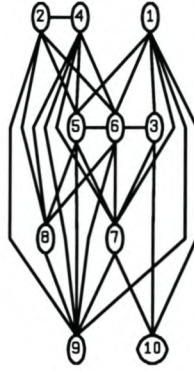


Figure A.1: The 10-node network

## The 20-node network

The 20-node network has 102 uni-directional links and carries 1 traffic class. The links are capacitated with 5,289,780 units of bandwidth. A total of 1,334,705 units of flow are offered to the 380 source-destination pairs.

## The 50-node network

The 50-node network has 202 uni-directional links and carries 1 traffic class. The links are capacitated with 38,519,240 units of bandwidth. A total of 6,581,372 units of flow are offered to the 2,450 source-destination pairs.

## The 100-node network

The 100-node network has 244 uni-directional links and carries 1 traffic class. The links are capacitated with 6,515,880 units of bandwidth. A total of 25,000 units of flow are offered to the 9,900 source-destination pairs.

## The planar 100-node network

The planar 100-node network has 470 uni-directional links and carries 1 traffic class. The links are capacitated with 1,787,100 units of bandwidth. A total of 25,000 units of flow are offered to the 9,900 source-destination pairs.

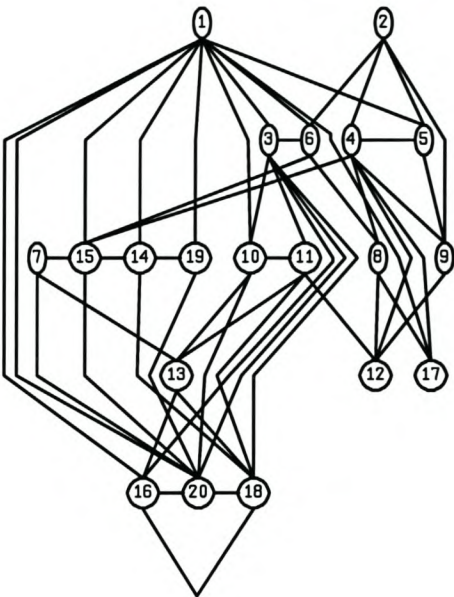


Figure A.2: The 20-node network

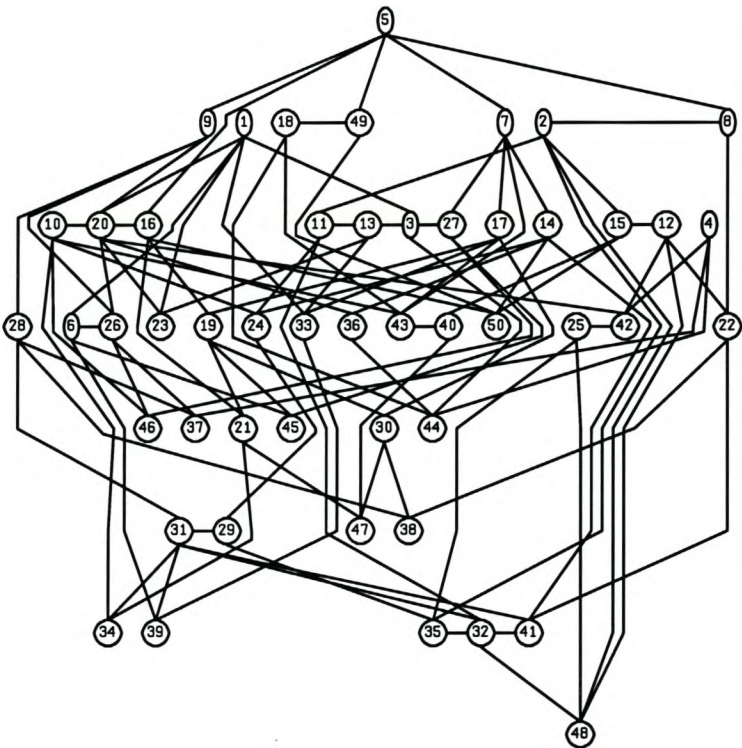


Figure A.3: The 50-node network



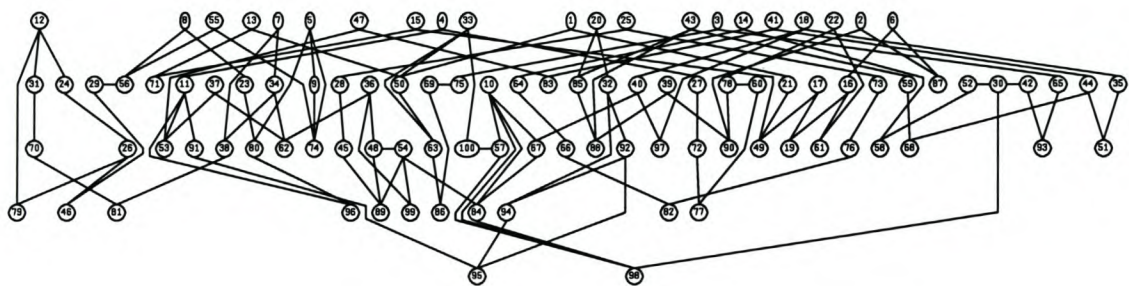


Figure A.4: The 100-node network

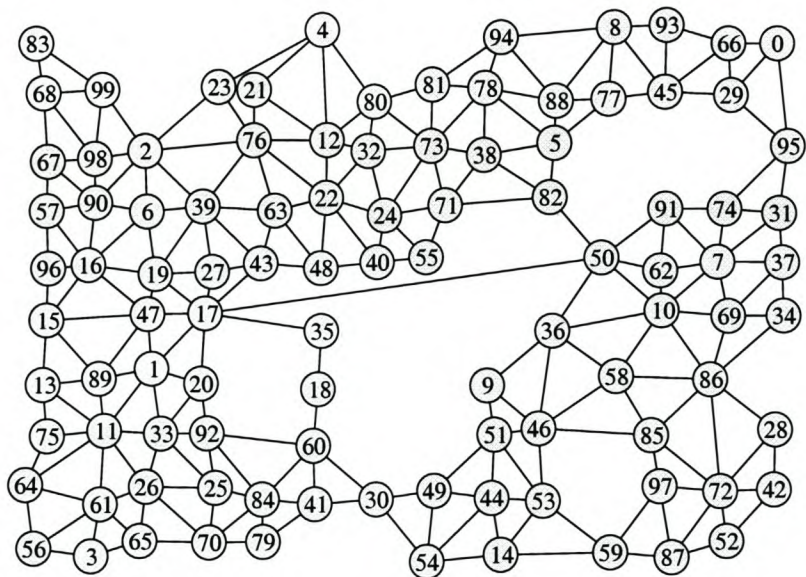


Figure A.5: The planar 100-node network

## Appendix B

# Route degeneracy

### B.1 Path sets and route degeneracy

The optimal solution  $\mathbf{B}$  computed by the flow deviation algorithm is not unique. For example consider the network presented in Fig. B.1 where traffic is offered from nodes 1 and 2 to node 6: the traffic demands are  $d_{(1,6)} = 0.5$  and  $d_{(2,6)} = 1.5$ . All links have capacity  $b_i = 2$  and have the same propagation delay and the same weight factor. The optimal link flows are

$$\begin{aligned} f_{(1,3)} &= 0.5 & f_{(2,3)} &= 1.5 \\ f_{(3,4)} &= 1.0 & f_{(3,5)} &= 1.0 \\ f_{(4,6)} &= 1.0 & f_{(5,6)} &= 1.0 \end{aligned}$$

Let  $f_P$  denote the flow on path  $P$ . We can assign any flow  $z$  where  $0 \leq z \leq 0.5$  to path  $(1, 3, 4, 6)$  whereupon the flows assigned to the other routes are

$$\begin{aligned} f_{(1,3,4,6)} &= z & f_{(1,3,5,6)} &= 0.5 - z \\ f_{(2,3,4,6)} &= 1.0 - z & f_{(2,3,5,6)} &= 0.5 + z. \end{aligned}$$

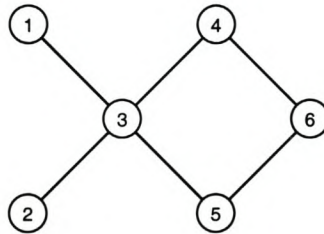


Figure B.1: The “fish” network



It is probably an advantage for a source-destination pair to have two paths rather than one path. Having four paths rather than three is probably a disadvantage. Operational requirements may prefer a particular value of  $z$ . Thus  $z = 0$  and  $z = 0.5$  will reduce the number of paths from four to three. The flow deviation algorithm yields  $z = 0.25$  which assigns two paths from each of nodes 1 and 2 to node 6 with equal bandwidth. From the point of view of robustness under traffic forecast error, this may be the preferred solution.

Given the link flows, we need methods to compute not only a set of paths and a set of path flows consistent with the link flows, but we also need criteria to determine which set of paths and path flows are superior, and we need mechanisms to find optimal (according to those criteria) path sets and path flows.

## Appendix C

# List of Acronyms

This appendix lists the acronyms used in this thesis.

ABR	Available Bit Rate
ACK	acknowledgement packet
ATM	Asynchronous Transfer Mode
BG	Bertsekas-Gallagher flow deviation algorithm
BGP	Border Gateway Protocol
B-ISDN	Broadband Integrated Services Digital Network
CR-LDP	Constraint-based Routing Label Distribution Protocol
CSPF	Constraint-based Shortest Path First
DLCI	Data Link Control Identifier
ER	Explicit Rate
FD	Flow Deviation
FEC	Forwarding Equivalence Class
G-MPLS	Generalized MPLS
GMPLS	Generalized MPLS
ICS	Iterative Capacity Splitting
IETF	Internet Engineering Task Force
ILP	Integer Linear Programming
IP	Internet Protocol
KSP	$k$ -shortest path



*Appendix C. Acronyms*

LDP	Label Distribution Protocol
LMP	Link Management Protocol
LP	linear programming
LSP	Label Switched Path
LSR	Label Switching Router
MCMCF	minimal cost multi-commodity flow
MPL(ambda)S	Multiprotocol Lambda Switching
MPLS	Multiprotocol Label Switching
NLP	nonlinear programming
OSPF	Open Shortest Path First
QoS	Quality of Service
RED	Random Early Detection
RSVP	ReSerVation Protocol
RSVP-TE	ReSerVation Protocol with Traffic Engineering extensions
SA	Simulated Allocation
SDH	Synchronous Digital Hierarchy
S-D	source-destination
SONET	Synchronous Optical Network
SPF	Shortest Path First
SRLG	Shared Risk Link Group
TCA	Thrifty Capacity Allocation
TCP	Transmission Control Protocol
TE	Traffic Engineering
TRACK	TRee-based ACKnowledgement
TRL	Traffic Restoration Level
VCI	Virtual Channel Identifier
VPI	Virtual Path Identifier
VP	Virtual Path
WDM	Wavelength Division Multiplexing

# Bibliography

- [1] Multiprotocol Label Switching White Paper. Future Software Private Limited, India, 1999.  
<http://www.futsoft.com>.
- [2] G. R. Ash. Traffic Engineering & QoS Methods for IP-, ATM-, & TDM-Based Multiservice Networks. Internet Draft, Work in Progress, draft-ietf-tewg-qos-routing-04.txt.pdf, Internet Engineering Task Force, October 2001. <http://www.research.att.com/~jrex/jerry/>.
- [3] M. J. Atallah, editor. *Algorithms and Theory of Computation Handbook*. CRC Press, 1999.
- [4] D. Awduche, A. Chiu, A. Elwalid, I. Widjaja, and X. Xiao. Overview and Principles of Internet Traffic Engineering. Internet Draft, Work in Progress, draft-ietf-tewg-principles-00.txt, Internet Engineering Task Force, August 2001.
- [5] D. Awduche, A. Hannan, and X. Xiao. Applicability Statement for Extensions to RSVP for LSP-Tunnels. Internet Draft, Work in Progress, draft-ietf-mpls-rsvp-tunnel-applicability-02.txt, Internet Engineering Task Force, April 2001.
- [6] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. McManus. Requirements for Traffic Engineering Over MPLS. RFC 2702, Internet Engineering Task Force, September 1999.
- [7] D. Awduche, Y. Rekhter, J. Drake, and R. Coltun. Multi-Protocol Lambda Switching: Combining MPLS Traffic Engineering Control with Optical Crossconnects. Internet Draft, Work in Progress, draft-awduche-mpls-te-optical-02.txt, Internet Engineering Task Force, July 2000.
- [8] B. A. Bagula. Traffic Engineering Label Switched Paths. Master's thesis, Department of Computer Science, University of Stellenbosch, 7600 Stellenbosch, South Africa, March 2002.
- [9] D. Bertsekas and R. Gallager. *Data Networks*. Prentice-Hall International, second edition, 1992.



## Bibliography

- [10] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin. Resource ReSerVation Protocol (RSVP) Version 1 Functional Specification. RFC 2205, Internet Engineering Task Force, September 1997.
- [11] J. E. Burns, T. J. Ott, J. M. de Kock, and A. E. Krzesinski. Path selection and bandwidth allocation in MPLS networks: a nonlinear programming approach. In *Proceedings of SPIE, ITCOM-2001*, volume 4523 of *Internet Performance and Control of Network Systems II*, pages 15–26, Denver, Colorado, USA, July 2001.
- [12] J. E. Burns, T. J. Ott, A. E. Krzesinski, and K. E. Müller. Path selection and bandwidth allocation in MPLS networks. *Performance Evaluation*, 52:133–152, 2003.
- [13] A. Busson, J. Rougier, and D. Kofman. Analysis and optimization of hierarchical reliable transport protocols. In *Proceedings of ITC-17 the Seventeenth International Teletraffic Congress*, 2000.
- [14] T. Carpenter, K. R. Krishnan, and D. Shallcross. Enhancements to Traffic Engineering for Multi Protocol Label Switching. In *Proceedings of ITC-17 the Seventeenth International Teletraffic Congress*, 2000.
- [15] T. Cinkler, P. Laborczi, and Á. Horváth. Protection through thrifty configuration. In *Proceedings of ITC-16 the Sixteenth International Teletraffic Congress*, pages 975–987. Elsevier Science B.V., 1999.
- [16] S. Cwlich, M. Deng, D. J. Houck, and D. F. Lynch. An LP-based approach to restoration network design. In *Proceedings of ITC-16 the Sixteenth International Teletraffic Congress*, pages 633–644. Elsevier Science B.V., 1999.
- [17] B. Davie, P. Doolan, and Y. Rekhter. *Switching in IP Networks: IP Switching, Tag Switching, and Related Technologies*, chapter 8. Morgan Kaufmann Publishers, 1998.
- [18] J. M. de Kock. Optimal Management of MPLS Networks. Master's thesis, Department of Computer Science, University of Stellenbosch, 7600 Stellenbosch, South Africa, March 2002.
- [19] R. Doverspike and J. Yates. Challenges for MPLS in Optical Network Restoration. In *IEEE Communications Magazine*, pages 89–96, February 2001.
- [20] D. Awduche et al. RSVP-TE: Extensions to RSVP for LSP Tunnels. Internet Draft, Work in Progress, draft-ietf-mpls-rsvp-lsp-tunnel-09.txt, Internet Engineering Task Force, August 2001.
- [21] D. Haskin et al. A Method for Setting Alternative Label Switched Paths to Handle Fast Reroute. Internet Draft, Work in Progress, draft-haskin-mpls-fast-reroute-05.txt, Internet Engineering Task Force, November 2000.

- [22] D. Papadimitriou et al. Inference of Shared Risk Link Groups. Internet Draft, Work in Progress, draft-many-inference-srlg-01.txt, Internet Engineering Task Force, July 2001.
- [23] J. Ash et al. LSP Modification Using CR-LDP. Internet Draft, Work in Progress, draft-ietf-mpls-crlsp-modify-03.txt, Internet Engineering Task Force, March 2001.
- [24] L. Cheng et al. A Framework for Internet Network Engineering. Internet Draft, Work in Progress, draft-cheng-network-engineering-framework-01.txt, Internet Engineering Task Force, July 2001.
- [25] V. Sharma et al. Framework for MPLS-based Recovery. Internet Draft, Work in Progress, draft-ietf-mpls-recovery-frmwk-03.txt, Internet Engineering Task Force, July 2001.
- [26] W. Lai et al. Network Hierarchy and Multilayer Survivability. Internet Draft, Work in Progress, draft-ietf-tewg-restore-hierarchy-00.txt, Internet Engineering Task Force, October 2001.
- [27] A. Farrel, P. Brittain, P. Matthews, and E. Gray. Fault Tolerance for LDP and CR-LDP. Internet Draft, Work in Progress, draft-ietf-mpls-ldp-ft-02.txt, Internet Engineering Task Force, May 2001.
- [28] D. Gan, P. Pan, A. Ayyangar, and K. Kompella. A Method for MPLS LSP Fast-Reroute Using RSVP Detours. Internet Draft, Work in Progress, draft-gan-fast-reroute-00.txt, Internet Engineering Task Force, April 2001.
- [29] A. Iwata, N. Fujita, G. Ash, and A. Farrel. Crankback Routing Extensions for MPLS Signaling. Internet Draft, Work in Progress, draft-iwata-mpls-crankback-01.txt, Internet Engineering Task Force, July 2001.
- [30] A. Iwata, N. Fujita, and T. Nishida. MPLS Signaling Extensions for Shared Fast Rerouting. Internet Draft, Work in Progress, draft-iwata-mpls-shared-fastreroute-00.txt, Internet Engineering Task Force, July 2001.
- [31] A. Kershenbaum. *Telecommunications Network Design Algorithms*, pages 252–330. McGraw Hill, 1993.
- [32] S. Kini, M. Kodialam, T. Lakshman, S. Sengupta, and C. Villamizar. Shared Backup Label Switched Path Restoration. Internet Draft, Work in Progress, draft-kini-restoration-shared-backup-01.txt, Internet Engineering Task Force, May 2001.
- [33] L. Kleinrock. *Queueing System Vol.2: Computer Applications*. John Wiley & Sons, New York, 1976.



- [34] A. E. Krzesinski and K. E. Müller. Traffic Engineering Label Switched Paths in IP Networks. In *Proceedings of SATNAC 2001 the Fourth Annual South African Telecommunications, Networks and Applications Conference*, Wild Coast Sun, South Africa, September 2001.
- [35] A. E. Krzesinski and K. E. Müller. Traffic protection in MPLS networks using a pre-planned flow optimization model. In *Proceedings of SPIE, ITCOM-2002*, volume 4865 of *Internet Performance and Control of Network Systems III*, pages 244–255, Boston, Massachusetts, USA, July 2002.
- [36] W. Lai, B. Christian, R. Tibbs, and S. Van den Berghe. A Framework for Internet Traffic Engineering Measurement. Internet Draft, Work in Progress, draft-ietf-tewg-measure-00.txt, Internet Engineering Task Force, August 2001.
- [37] J. Lang and B. Rajagopalan et al. Generalized MPLS Recovery Functional Specification. Internet Draft, Work in Progress, draft-bala-gmpls-recovery-functional-00.txt, Internet Engineering Task Force, August 2002.
- [38] J. Lang and K. Mitra et al. Link Management Protocol (LMP). Internet Draft, Work in Progress, draft-ietf-mpls-lmp-02.txt, Internet Engineering Task Force, September 2001.
- [39] I. Lemke and G. Sander. VCG Visualization Tool version 1.3 – Visualization of Compiler Graphs. University of Saarland, Germany. Obtainable from <http://www.cs.uni-sb.de/RW/users/sander/html/gsvcg1.html>.
- [40] Linear problem solver: lp\_solve version 3.0, September 1999. Obtainable from [ftp://ftp.ics.ele.tue.nl/pub/lp\\_solve/](ftp://ftp.ics.ele.tue.nl/pub/lp_solve/).
- [41] K. Murakami and H. S. Kim. Virtual Path Routing for Survivable ATM Networks. In *IEEE/ACM Transactions on Networking*, volume 4, pages 22–39, February 1996.
- [42] T. J. Ott. A Multi-Commodity Flow Problem related to MPLS. Telcordia Technologies, Inc., 445 South Street, Morristown, NJ 07960, USA, June 2000.
- [43] T. J. Ott, T. Carpenter, D. Shallcross, T. Bogovic, and K. R. Krishnan. Algorithms for Flow Allocation for Multi Protocol Label Switching. Telcordia Technologies, Inc., 445 South Street, Morristown, NJ 07960, USA, May 2000.
- [44] K. Owens, V. Sharma, and M. Oommen. Network Survivability Considerations for Traffic Engineered IP Networks. Internet Draft, Work in Progress, draft-owens-te-network-survivability-01.txt, Internet Engineering Task Force, July 2001.

- [45] M. Pióro and M. Szcześniak. Routing of disjoint backup paths in ATM networks. In *Proceedings of IFIP*, pages 162–175. Chapman & Hall, 1997.
- [46] M. Pióro and T. Szymański. Basic reconfiguration options in multi-layer robust telecommunication networks – design and performance issues. In *Proceedings of ITC-17 the Seventeenth International Teletraffic Congress*, 2000.
- [47] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C*. Cambridge University Press, second edition, 1992.
- [48] E. Rosen, A. Viswanathan, and R. Callon. Multiprotocol Label Switching Architecture. RFC 3031, Internet Engineering Task Force, January 2001.
- [49] P. A. Veitch, I. Hawker, and D. G. Smith. Enhancing the self-healing capability of statically protected ATM networks. In *Proceedings of IFIP*, pages 141–161. Chapman & Hall, 1997.
- [50] C. Villamizar. <http://brookfield.ans.net/omp/random-test-cases.html>.

Note: Internet Engineering Task Force (IETF) Internet Drafts and Request for Comments (RFC) documents are obtainable from <http://www.ietf.org/lid-abstracts.txt>.